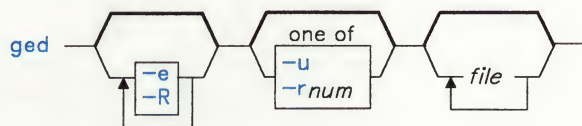

ged

Purpose

Displays, makes, and edits graphical files on Tektronix 4010 terminals.

Syntax



OL777037

Description

The **ged** command is an interactive graphical editor used to edit drawings on Tektronix 4010 series display terminals. The drawings are a sequence of objects that consist of **lines**, **arcs**, and **text**. With **ged** you can view the objects at various magnifications and from various locations. The drawings are stored in graphics primitive string (**GPS**) files. If you specify - (minus) as the file name, **ged** reads standard input into the edit buffer.

An arc or lines object has a start point (**object-handle**), followed by zero or more points (**point-handles**). A text object has only an object-handle. These objects are positioned within a Cartesian plane (**universe**), having 64K (-32K to +32K) points (**universe-units**) on each axis. The **GPS** universe is divided into 25 equal sized areas called **regions**. These regions are arranged in five rows of five squares each, numbered 1 to 25 from the lower left of the universe to the upper right.

The **ged** command maps rectangular areas (**windows**) from the universe onto the display screen. Windows let you view pictures from different locations and at different magnifications. The **universe-window** is the window with minimum magnification; that is, the window that views the entire universe. The **home-window** is the window that completely displays the contents of the display buffer.

Flags

- e Does not erase the screen before the initial display
- rnum Displays region number *num*.
- u Displays the entire **GPS** universe.
- R Invokes the restricted shell on use of ! (exclamation character).

Subcommands

The **ged** subcommands are entered in **stages**. Typically each stage ends with a **<cr>** (Return). Prior to the final **<cr>**, you may cancel the subcommand by pressing **INTERRUPT (Alt-Pause)**. You can edit the input of a stage, during the stage, by using the erase and kill characters of the calling shell. The ***** (star) prompt indicates that **ged** is waiting at stage 1.

Each subcommand consists of a subset of the following stages:

1. **Command line**, whose format is the same as the format of a shell command:

subcommand-name [-flags] [filename]

followed by pressing the **Enter** key. The *subcommand-name* consists of the first character of the subcommand. **ged** echoes the full command name and pauses for the remainder of the command line. Flags are indicated by a leading - (minus). To generate a list of **ged** subcommands, enter: **?**.

2. **Text**, a sequence of characters terminated by an unescaped **Enter**. You can have a maximum of 120 lines of text.
3. **Points**, a sequence of one or more screen locations (maximum of 30), indicated either by the terminal cross hairs or by name. The prompt for entering points is the appearance of the cross hairs. When the cross hairs are visible, type:

sp (space)	Enters the current location as a point. The point is identified by a number.
\$num	Enters the previous point numbered <i>num</i> .
> x	Labels the last point entered with the upper case letter <i>x</i> .
\$x	Enters the point labeled <i>x</i> .
.	Establishes the previous points as the current points. At the start of a command, the previous points are those locations given with the previous command.
=	Echoes the current points.
\$.num	Enters the point.
#	Erases the last point entered.
@	Erases all of the points entered.

4. **Pivot**, a single location entered by pressing the **Enter** key or by using the **\$** operator and indicated with a ***** (star).
5. **Destination**, a single location entered by pressing the **Enter** key or by using **\$** (dollar sign).

Subcommand Summary

In the following lists, characters printed in **bold** are to be entered literally. Subcommand stages are printed in ***bold italics***. Arguments surrounded by [] (brackets) are optional. Parentheses surrounding arguments separated by “or” indicate that you must specify exactly one of the arguments.

Construct Subcommands

Arc	[-echo,style,weight] <i>points</i>
Box	[-echo,style,weight] <i>text</i>
Circle	[-echo,style,weight] <i>point</i>
Hardware	[-echo] <i>text points</i>
Lines	[-echo,style,weight] <i>points</i>
Text	[-angle,echo,height, mid-point,right-point,text, weight] <i>text points</i>

Edit Subcommands

Delete	(-(universe or view) or <i>points</i>)
Edit	[-angle,echo,height,style,weight] (-(universe or view) or <i>points</i>)
Kopy	[-echo,points,x] <i>points pivot destination</i>
Move	[-echo,points,x] <i>points pivot destination</i>
Rotate	[-angle,echo,kopy,x] <i>points pivot destination</i>
Scale	[-echo,factor,kopy,x] <i>points pivot destination</i>

View Subcommands

coordinates	<i>points</i>
erase	
new-display	
object-handles	(-(universe or view) or <i>points</i>)
point-handles	(-(labelled-points or universe or view) or <i>points</i>)
view	(-(home or universe or region) or [-x] <i>pivot destination</i>)
x	[-view] <i>points</i>
zoom	[-out] <i>points</i>

Other Subcommands

quit or **Quit**

read [-angle,echo,height, mid-point,right-point,text, weight] *file-name*[*destination*]

set [-angle,echo,factor, height,kopy,mid-point,points, right-point,style,text, weight,x]

write *file-name*

!*command*

?

Options

Options specify parameters used to construct, edit, and view graphical objects. If a parameter used by a subcommand is not specified as an *option*, the default value for the parameter will be used (see set following). The format of subcommand *options* is:

-*option*[,*option*]

where *option* is *keyletter*[*value*]. Flags take on the *values* of true or false indicated by + and - respectively. If no *value* is given with a flag, true is assumed.

Object Options

anglen	Specifies an angle of <i>n</i> degrees.										
echo	When true, changes made to the display buffer are echoed to the screen.										
factorn	Specifies a scale factor is <i>n</i> percent.										
heightn	Sets the height of text to <i>n</i> universe-units ($0 \leq n < 1280$).										
kopy	When true, copies rather than moves.										
mid-point	When true, uses the mid-point of a text string to locate string.										
points	When true, operates on points; otherwise operates on objects.										
right-point	When true, uses the rightmost point of the text string to locate string.										
styletype	Sets the line style to one of following <i>types</i> : <table><tr><td>so</td><td>solid</td></tr><tr><td>da</td><td>dashed</td></tr><tr><td>dd</td><td>dot-dashed</td></tr><tr><td>do</td><td>dotted</td></tr><tr><td>ld</td><td>long-dashed.</td></tr></table>	so	solid	da	dashed	dd	dot-dashed	do	dotted	ld	long-dashed.
so	solid										
da	dashed										
dd	dot-dashed										
do	dotted										
ld	long-dashed.										
text	When false, outlines rather than draws text strings.										

weighttype Sets line weight to one of following *types*:

n	narrow
m	medium
b	bold.

Area Options

home	References the home-window.
out	Reduces magnification during zoom.
region <i>n</i>	References the region <i>n</i> .
universe	References the universe-window.
view	References those objects currently in view.
x	Indicates the center of the referenced area.

Subcommand Descriptions

Construct Subcommands

Arc

Lines Behave similarly. Each consists of a *command line* followed by *points*. The first *point* entered is the object-handle. Successive *points* are point-handles. **Lines** connects the handles in numerical order. **Arc** fits a curve to the handles (currently a maximum of 3 points will be fit with a circular arc; splines will be added in a later version).

Box

Circle

Special cases of **Lines** and **Arc**, respectively. **Box** generates a rectangle with sides parallel to the universe axes. A diagonal of the rectangle would connect the first *point* entered with the last *point*. The first *point* is the object-handle. Point-handles are created at each of the vertices. **Circle** generates a circular arc centered about the *point* numbered zero and passing through the last *point*. The circle's object-handle coincides with the last *point*. A point-handle is generated 180 degrees around the circle from the object-handle.

Text

Hardware

Generate *text* objects. Each consists of a *command line*, *text* and *points*. *Text* is a sequence of characters delimited by <cr>. Multiple lines of text may be entered by preceding a **cr** with a \ (backslash). The **Text** subcommand creates software-generated characters. Each line of software text is treated as a separate *text* object. The first *point* entered is the object-handle for the first line of text. The **Hardware** command sends the characters in *text*, uninterpreted, to the terminal.

Edit Subcommands

Edit subcommands operate on portions of the display buffer called ***defined-areas***. A defined-area is referenced either with an area *option* or interactively. If an area *option* is not given, the perimeter of the defined-area is indicated by ***points***. If no ***point*** is entered, a small defined-area is built around the location of the **<cr>**. This is useful to reference a single ***point***. If only one ***point*** is entered, the location of the **<cr>** is taken in conjunction with the ***point*** to indicate a diagonal of a rectangle. A defined-area referenced by ***points*** will be outlined with dotted lines.

- | | |
|---------------|--|
| Delete | Removes all objects whose object-handle lies within a defined-area. The universe option removes all objects and erases the screen. |
| Edit | Modifies the parameters of the objects within a defined-area. Parameters that can be edited are:

angle Specifies the angle of <i>text</i>
height Specifies the height of <i>text</i>
style Specifies the style of <i>lines</i> and <i>arc</i>
weight Specifies the weight of <i>lines</i> , <i>arc</i> , and <i>text</i> |
| Kopy | |
| Move | Copies (or moves) object- and/or point-handles within a defined-area by the displacement from the <i>pivot</i> to the <i>destination</i> . |
| Rotate | Rotates objects within a defined-area around the <i>pivot</i> . If the kopy flag is true, the objects are copied rather than moved. |
| Scale | For object whose object-handles are within a defined-area, point displacements from the <i>pivot</i> are scaled by factor percent. If the kopy flag is true then the objects are copied rather than moved. |

View Subcommands

- | | |
|-----------------------|--|
| coordinates | Displays the location of <i>point</i> (s) in universe- and screen-units. |
| erase | Clears the screen (but not the display buffer). |
| new-display | Erases the screen; then displays the display buffer. |
| object-handles | |
| point-handles | Labels object- (and/or point-handles) that lie within the defined-area with O (or P). point-handles identifies labeled points when the labeled-points flag is true. |
| view | Moves the window so that the universe point corresponding to the <i>pivot</i> coincides with the screen point corresponding to the <i>destination</i> .
Options for home , universe , and region display particular windows in the universe. |

x	Indicates the center of a defined-area. Option view indicates the center of the screen.
zoom	Decreases (zoom out) or increases the magnification of the viewing window based on the defined-area. For increased magnification, the window is set to circumscribe the defined-area. For a decrease in magnification the current window is inscribed within the defined-area.

Other Subcommands

quit	
Quit	Exit from ged . quit responds with ? if the display buffer has not been written since the last modification.
read	Inputs the contents of a file. If the file contains a GPS object, it is read directly. If the file contains text it is converted into <i>text</i> object(s). The first line of a text file begins at destination .
set	When given <i>option</i> (s) resets default parameters; otherwise it prints current default values.
write	Outputs the contents of the display buffer to a file.
!	Escapes ged to execute an AIX Operating System command.
?	Lists ged subcommands.

Related Information

The following commands: “**gdev**” on page 460, “**graphics**” on page 497, and “**sh**” on page 913.

The **GPS** file in *AIX Operating System Technical Reference*.

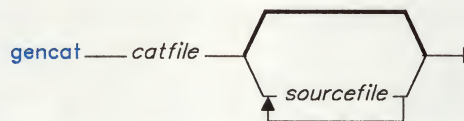
gencat

gencat

Purpose

Creates and modifies a message catalog.

Syntax



OL805484

Description

Use **gencat** to create a message catalog from a message text source file. Because **gencat** conforms to X/Open specifications, **gencat** does not accept symbolic identifiers. You must run **mkcatdefs** as described in “Symbolic Message Identifiers” on page 474 if you want to use symbolic identifiers.

The format for **gencat** is:

```
$ gencat catfile [sourcefiles]
```

If a message catalog with the name *catfile* exists, **gencat** will modify it according to the statements in the message source files. If it does not exist, **gencat** creates a catalog file with the name *catfile*.

You may specify any number of message text source files. **gencat** processes multiple source files one after the other in the sequence that you specify them. Each successive source file will modify the catalog. If you do not specify a source file, **gencat** accepts message source data from standard input.

A message text source file is a text file that you create to enter messages into. You can use any text editor to enter the messages. Assign message set numbers and message ID numbers to each message by using the commands described in this section.

Use the **\$set** command in a source file to give a group of messages a set number. The format of the **\$set** command is:

```
$set n [comment]
```

The message set number is specified by *n*. All messages following the **\$set** command are assigned that set number, up until the next occurrence of a **\$set** command. You must specify at least one set number if the source file contains message text. The set numbers must be assigned in ascending order, but need not be contiguous. Large gaps in the number sequence do not affect performance, but will increase the size of the catalog. There is no performance advantage to using more than one set number in a catalog. Set numbers must start at 1. The highest set number in a X/Open-conforming application is 255.

You may include a comment in the **\$set** command, but it is not required. The following example includes a comment:

```
$set 10  Communication Error Messages
```

Use the **\$delset** command to remove all of the messages belonging to the specified set from a catalog. The format of the **\$delset** command is:

```
$delset n [comment]
```

The message set is specified by *n*. The **\$delset** command must be placed in the proper set number order with respect to any **\$set** commands in the same source file. You may include a comment in the **\$delset** command also.

You may include a comment line anywhere in the source file, except within message text. Indicate comments as shown below:

```
$ [comment]
```

You must leave at least one space after the **\$**.

Enter the message text and assign message ID numbers as follows:

```
m message-text
```

This assigns the message ID number *m* to the text that follows it. You must leave at least one space after the message number.¹ Message numbers must be in ascending order within a single message set, but need not be contiguous. Large gaps in the number sequence do not affect performance, but will increase the size of the catalog. In an X/Open-conforming application, message numbers must be in the range 1-32767.

All text following the message number is included as message text, up to the end of the line. Use the escape character **** to continue message text on the following line. The **** must be the last character on the line. Consider the following example:

```
5 This is the text associated with \  
message number 5.
```

¹ AIX allows any amount of white space after the message ID number; however X/Open specifies that you leave only one space between the message number and the message text.

These two lines define the single-line message:

This is the text associated with message number 5.

The escape character `\` may be used to include special characters in the message text. These special characters are defined as follows:

- `\n` Performs a new-line function when the message is displayed.
- `\t` Inserts a horizontal tab character when the message is displayed.
- `\v` Inserts a vertical tab when the message is displayed.
- `\b` Performs a backspace function when the message is displayed.
- `\r` Inserts a carriage return character when the message is displayed.
- `\f` Inserts a form feed character when the message is displayed.
- `\\` Displays the `\` (backslash) character in the message.
- `\ddd` Displays the single-byte character associated with the octal value represented by the valid octal digits *ddd*. One, two, or three octal digits may be specified; however you must include leading zeros if the characters following the octal digits are also valid octal digits. For example, the octal value for `$` is 44. To display `$5.00` use `\0445.00`, not `\445.00`, or the 5 will be parsed as part of the octal value.
- `\xddd2` Displays the single-byte or double-byte character associated with the hexadecimal value represented by the four valid hexadecimal digits *dddd*. You may specify one, two, three, or four digits, but you must include leading zeros to avoid parsing errors (see `\ddd`).

You can also include **printf** conversion specifications in messages that are displayed by applications using **printf** or **NLprintf** (see **printf** in *AIX Operating System Technical Reference*). If you display a message from a shell script with **dspmsg**, the message can contain the `%s` or `%n$s` conversion specifications (see “**dspmsg**” on page 359).

You can use the **\$quote** command in a message source file to define a character for delimiting message text. The format for this command is:

\$quote [*char*] [*comment*]

Use the specified character before and after the message text as shown in the following example source file:

² This escape sequence is an AIX extension to X/Open specifications.

`$quote "` Use a double quote to delimit message text

`$set 10` Message Facility - Quote command messages

1 "Use the `$quote` command to define a character \

\n for delimiting message text"

2 "You can include the \"quote\" character in a message \n \

by placing a \\ in front of it"

3 You can include the "quote" character in a message \n \

by having another character as the first nonblank \

\n character after the message ID number

`$quote`

4 You can disable the quote mechanism by \n \

using the `$quote` command without \n a character \

after it

In this example, the `$quote` command defines the double quote (") as the **quote** character. The quote character must be the first non-blank character following the message number. Any text following the next occurrence of the quote character is ignored.

The example also shows two ways the quote character can be included in the message text:

- Place a \ in front of the quote character.
- Use some other character as the first non-blank character following the message number. This disables the quote character only for that message.

The example shows these other things:

- A \ is still required to split a quoted message across lines.
- To display a \ in a message you must place another \ in front of it.
- You can format your message with a new-line character by using \n.
- If you use the `$quote` command with no character argument, you disable the quote mechanism.

After entering your messages into a source file you must use the message facility program **gencat** to process the source file to create a message catalog.

Symbolic Message Identifiers

AIX provides a mechanism that allows symbolic references to messages by letting you use alphanumeric identifiers instead of set numbers and message ID numbers.³ You assign the identifiers to sets and messages in the source file in the same manner that you assign set numbers and message ID numbers.

The symbolic identifiers can contain ASCII letters, digits, and underscores. The first character cannot be a digit. The maximum length cannot exceed 64 bytes.

The following example shows a message source file with symbolic message identifiers:

```
$set symbolic      Message Facility - Symbolic ID's
$quote *
```

```
ID_names  *Symbolic identifier syntax: \n \
\talphanumerics or underscores \n \
\tnon-digit first character \n \
\t64 byte maximum length *
```

```
set_use   *To assign set ID: \n \
\t$set "identifier" [comment] *
```

```
msg_use   *To assign message ID: \n \
\t"identifier" message-text *
```

Japanese Language Support Information

If Japanese Language Support is installed on your system, this command is not available.

Related Information

The following commands: “**dspcat**” on page 357, “**dspmsg**” on page 359, “**mkcatdefs**” on page 651, and “**runcat**” on page 852.

The **catopen**, **catgets**, **catgetamsg**, **catclose**, **NLcatopen**, **NLcatgets**, and **NLgetamsg** files in *AIX Operating System Technical Reference*.

The discussion of **gencat** in *AIX Operating System Programming Tools and Interfaces*.

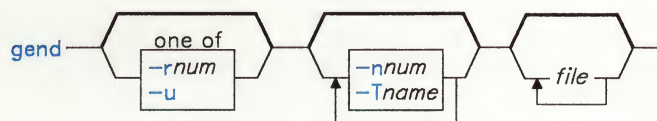
³ Symbolic references are not defined by the X/Open specification. This mechanism is an AIX extension.

gend

Purpose

Provides a general graphics device backend.

Syntax



OL805458

Description

The `/usr/bin/graf/gend` command displays **GPS** files on the graphics output devices supported by the Advanced Display Graphics Support Library (GSL). For more information about GSL, see the “Advanced Display Graphics Support Library” in *AIX Operating System Technical Reference*. By default, **gend** reads standard input and writes to the current display (see “**display**” on page 332), but **gend** can also drive printers and plotters if you have installed the VDI drivers that are in the Extended Services Program. You can specify the name of one or more **GPS** files on the command line. If you enter a file name of - (minus), **gend** reads standard input.

When **gend** displays an image, it opens a new virtual terminal. You can move to and from this virtual terminal by pressing Next Window (**Alt-Action**). See “**open**” on page 728 and *Using the AIX Operating System* for information on virtual terminals. To end **gend** and close the virtual terminal, press END OF FILE (**Ctrl-D**).

Note: The **gend** command produces the standard **GPS** line style attributes with one exception. Line style 4 (long dashed) is rendered as dash-dot-dot.

Flags

-nnum Specifies the number of chords per circle. Legal values are **64**, **128**, **256**, or **512**. The default value is **128**.

Rather than drawing truly circular arcs or circles, **gend** converts them into a series of very short line segments (**chords**), whose end points lie on the circle. For most devices and images, the default value of 128 is satisfactory. The higher values give a smoother image; the lower value provides faster drawing time.

gend

- rnum** Displays data in **GPS** region *num*. A **GPS** object is defined in a Cartesian plane of 64K points on each axis. The plane, or universe, is divided into 25 square regions numbered 1 to 25 from the lower left to the upper right.
- u** Displays data in the entire **GPS** universe.
- Tname** Uses the device specified by the *name* environment variable. The default is the current display (this must be supported by **/dev/hft**). When the image is to be displayed on other devices, you must ensure that the proper VDI device handler is installed.

Files

/usr/bin/graf/gend	The general devices backend.
/tmp/dev.XXXXXXX	Temporary file.

Related Information

The following commands: “**ged**” on page 463, “**gdev**” on page 460, “**graphics**” on page 497, and “**open**” on page 728.

“Advanced Display Graphics Support Library” in *AIX Operating System Technical Reference*

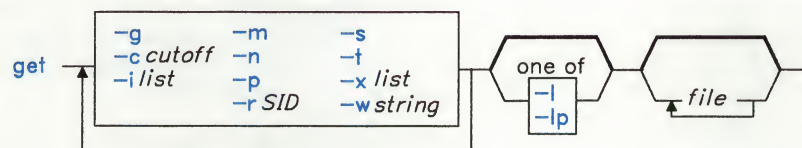
Installing programs in *Installing and Customizing the AIX Operating System*.

get

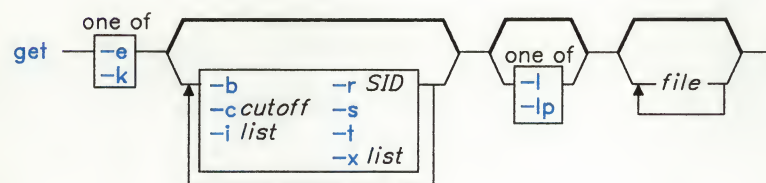
Purpose

Creates a specified version of a Source Code Control System (SCCS) file.

Syntax



OL805058



OL805355

Description

The **get** command reads the specified versions of the named Source Code Control System (SCCS) files, creates an ASCII text file for each *file* according to the specified flags, and writes each text file to a file with the same name as the original SCCS file without the *s.* (s period) prefix (the *g-file*). The flags and *files* can be specified in any order, and all flags apply to all named files.

If you specify a directory in place of *file*, **get** performs the requested actions on all the files in the directory that begin with the *s.* prefix. If you specify a - (minus) in place of a *file*, **get** reads standard input and interprets each line as the name of an SCCS file. **get** continues to read input until it reads END OF FILE (Ctrl-D).

If the effective user has write permission in the directory containing the SCCS files but the real user does not, then only one file can be named when the `-e` flag is used.

If you are not familiar with the terms *SID* and *delta* or you do not know the numbering system of the deltas, see *AIX Operating System Programming Tools and Interfaces* for more information.

SCCS Files

In addition to the file with the **s.** prefix (the **s-file**), **get** can create several auxiliary files: the **g-file**, **l-file**, **p-file**, and **z-file**. These files are identified by their **tag**, the letter before the hyphen. **get** names auxiliary files by replacing the leading **s.** in the SCCS file name with the proper tag, except for the g-file, which is named by removing the **s.** prefix. So, for a file named **s.sample**, the auxiliary file names would be **sample**, **l.sample**, **p.sample**, and **z.sample**.

These files serve the following purposes:

s-file This file contains the original file text and all the changes (**deltas**) made to the file. It also includes information about who can change the file contents, who has made changes, when those changes were made, and what the changes were. You cannot edit this file directly since the file is read-only. It contains the information needed by the SCCS commands to build the g-file, the file you can edit.

g-file The g-file is an ASCII text file that contains the text of the SCCS file version that you specify with the **-r** flag (or the latest trunk version by default). You can edit this file directly. When you have made all your changes and you want to make a new delta to the file, you can then apply the **delta** command to the file. **get** creates the g-file in the current directory.

The **get** command creates a g-file whenever it runs, unless the **-g** flag or the **-p** flag is specified. The real user owns it (not the effective user). If you do not specify the **-k** or the **-e** flag, the file is read-only. If the **-k** or the **-e** flag is specified, the owner has write permission for the g-file. You must have write permission in the current directory to create a g-file.

l-file The **get** command creates the l-file when the **-l** flag is specified. The l-file is a read-only file. It contains a table showing which deltas were applied in generating the g-file. You must have write permission in the current directory to create an l-file. Lines in the l-file have the following format:

1. A blank character if the delta was applied; a ***** appears otherwise.
2. A blank character if the delta was applied or was not applied and ignored; a ***** appears if the delta was not applied and was not ignored.
3. A code indicating a special reason why the delta was or was not applied:

Blank Included or excluded normally.

I Included using the **-i** flag.

X Excluded using the **-x** flag.

C Cut off using the **-c** flag.

4. The SID.
5. The date and time the file was created.
6. The login name of person who created the delta.

Comments and MR data follow on subsequent lines, indented one horizontal tab character. A blank line ends each entry.

For example, for a delta cutoff with the **-c** flag, the entry in the l-file might be:

```
**C 1.3 85/03/13 12:44:16 pat
```

and the entry for the initial delta might be:

```
1.1 85/02/27 15:42:20 pat
date and time created 85/02/27 15:42:20 by pat
```

p-file The **get** command creates the p-file when the **-e** or the **-k** flag is specified. The p-file passes information resulting from a **get -e** to a **delta** command. The p-file also prevents a subsequent execution of **get** with a **-e** flag for the same SID until **delta** is run or the joint edit key letter (**j**) is set in the SCCS file. The **j** key letter allows several **gets** on the same SID. The p-file is created in the directory containing the SCCS file. To create a p-file in the SCCS directory, you must have write permission in that directory. The permission code of the p-file is read-only to all but its owner, and it is owned by the effective user. The p-file contains:

- The current SID
- The SID of new delta to be created
- The user name
- The date and time of the **get**
- The **-i** flag, if it was present
- The **-x** flag, if it was present.

The p-file contains an entry with the preceding information for each pending delta for the file. No two lines have the same new delta SID.

z-file The z-file is a lock mechanism against simultaneous updates. The z-file contains the binary process number of the **get** command that created it. It is created in the directory containing the SCCS file and exists only while the **get** command is running.

When you use the **get** command, it displays the SID being accessed and the number of lines created from the SCCS file. If you specify the **-e** flag, the SID of the delta to be made appears after the SID is accessed and before the number of lines created. If you specify more than one file, or a directory, or standard input, **get** displays the file name before each file is processed. If you specify the **-i** flag, **get** lists included deltas below the word Included. If you specify the **-x** flag, **get** lists excluded deltas below the word Excluded.

Identification Keywords

You can use identification keywords in your files to insert identifying information. These keywords are replaced by their values in the g-file when **get** is invoked without the **-e** or **-k** flag. The following identification keywords can be used in SCCS files:

- %M%** Module name: the value of the **m** flag in the SCCS file.
- %I%** The SID (%R%.%L%.%B%.%S%) of the g-file.
- %R%** Release.
- %L%** Level.
- %B%** Branch.
- %S%** Sequence.
- %D%** Date of the current **get** (YY/MM/DD).
- %H%** Date of the current **get** (MM/DD/YY).
- %T%** Time of the current **get** (HH:MM:SS).
- %E%** Date newest applied delta was created (YY/MM/DD).
- %G%** Date newest applied delta was created (MM/DD/YY).
- %U%** Time newest applied delta was created (HH:MM:SS).
- %Y%** Module type: the value of the **t** flag in the SCCS file.
- %F%** SCCS file name.
- %P%** Full path name of the SCCS file.
- %Q%** The value of the **q** flag in the file.
- %C%** The current line number. This keyword is intended for identifying messages output by the program. It is not intended to be used on every line to provide sequence numbers.
- %Z%** The 4-character string @(#) recognized by the **what** command.
- %W%** A shorthand notation for constructing **what** strings for AIX program files. Its value is the characters and key letters:
%W% = %Z%%M%<horizontal-tab>%I%
- %A%** Another shorthand notation for constructing **what** strings for non-AIX program files. Its value is the key letters:
%A% = %Z%%Y% %M% %I%%Z%

The following table illustrates how **get** determines the SID of the file it retrieves, and what the pending SID is. The column **SID Specified** shows the various ways the SID can be specified with the **-r** flag. The two columns illustrate the various conditions that can exist, including whether or not the **-b** flag is used with the **get -e**. The **SID Retrieved** indicates the SID of the file that makes up the g-file. The **SID of Delta to be Created** column indicates the SID of the version that will be created when **delta** is applied.

SID Specified	-b Used	Other Conditions	SID Retrieved	SID of Delta to be Created
none ¹	no	R defaults to mR ²	mR.mL	mR.(mL + 1)
none ¹	yes	R defaults to mR	mR.mL	mR.mL.(mB + 1).1
(R)elease	no	R > mR	mR.mL	R.1 ³
R	no	R = mR	mR.mL	mR.(mL + 1)
R	yes	R > mR	mR.mL	mR.mL.(mB + 1).1
R	yes	R = mR	mR.mL	mR.mL.(mB + 1).1
R	N/A	R < mR and R does not exist	hR.mL ⁴	hR.mL.(mB + 1).1
R	N/A	R < mR and R exists	R.mL	R.mL.(mB + 1).1
R.(L)evel	no	No trunk successor	R.L	R.(L + 1)
R.L	yes	No trunk successor	R.L	R.L(mB + 1).1
R.L	N/A	Trunk successor in release \geq R	R.L	R.L.(mB + 1).1
R.L.(B)ranch	no	No branch successor	R.L.B.mS	R.L.B.(mS + 1)
R.L.B	yes	No branch successor	R.L.B.mS	R.L.(mB + 1).1
R.L.B.(S)equence	no	No branch successor	R.L.B.S	R.L.B.(S + 1)
R.L.B.S	yes	No branch successor	R.L.B.S	R.L.(mB + 1).1
R.L.B.S	N/A	Branch successor	R.L.B.S	R.L.(mB + 1).1

¹ Applies only if the **d** (default SID) flag is not present in the file (see “**admin**” on page 41).
² The mR indicates the maximum existing release.
³ Forces creation of the first delta in a new release.
⁴ The hR is the highest existing release that is lower than the specified, nonexistent, release R.

Figure 2. SID Determination

Flags

- b** Specifies that the delta to be created should have an SID in a new branch. The new SID is numbered according to the rules stated in Figure 2. You can use **-b** only with the **-e** flag. It is only necessary when you want to branch from a **leaf delta** (a delta without a successor). Attempting to create a delta at a nonleaf delta automatically results in a branch, even if the **b** header flag is not set. If you do not specify the **b** header flag in the SCCS file, **get** ignores the **-b** flag because the file does not allow branching (see the discussion of header flags on page 44).

-ccutoff Specifies a *cutoff* date and time, in the form: `YY[MM[DD[HH[MM[SS]]]]]` **get** includes no deltas to the SCCS file created after the specified *cutoff* in the g-file. The values of any unspecified items in the *cutoff* default to their maximum allowable values. Thus, a cutoff date and time specified with only the year (YY) would specify the last month, day, hour, minute, and second of that year. Any number of nonnumeric characters can separate the two-digit items of the *cutoff* date and time. This allows you to specify a date and time in a number of ways, as follows:

```
-c85/9/2,9:00:00
-c"85/9/2 9:00:00"
"-c85/9/2 9:00:00"
```

-e Indicates that the g-file being created is to be edited by the user applying **get**. The changes are recorded later with the **delta** command. **get -e** creates a p-file that prevents other users from issuing another **get -e** and editing a second g-file on the same SID before **delta** is run. The owner of the file can override this restriction by allowing joint editing on the same SID through the use of the **admin** command with the **-fj** flag. Other users, with permission, can obtain read-only copies by using **get** without the **-e** flag. The **get -e** command enforces SCCS file protection specified via the ceiling, floor, and authorized user list in the SCCS file (see "**admin**" on page 41).

-g Suppresses the actual retrieval of text from the SCCS file. Use the **-g** flag primarily to create an l-file or to verify the existence of a particular SID. Do not use it with the **-e** flag.

-ilist Specifies a *list* of deltas to be included in the creation of a g-file. The **SID list format** consists of a combination of individual SIDs separated by commas and SID ranges indicated by two SIDs separated by a hyphen. You specify the same SIDs with both the following command lines:

```
get -e -il.4,1.5,1.6 s.file
get -e -il.4-1.6 s.file
```

You can specify the SCCS identification of a delta in any form shown in the **SID Specified** column of Figure 2 on page 481. **get** interprets partial SIDs as shown in the **SID Retrieved** column of the table.

-k Suppresses replacement of identification keywords in the g-file by their value (see "Identification Keywords" on page 480). The **-k** flag is implied by the **-e** flag. If you accidentally ruin the g-file created by **get** with an **-e** flag, you can recreate it by reissuing the **get** command with the **-k** flag in place of the **-e** flag.

-l[p] Writes a delta summary to an l-file. If you specify **-lp**, the delta summary is written to standard output, and **get** does not create the l-file. Use this flag to determine which deltas were used to create the g-file currently in use. See "SCCS Files" on page 478 for the format of the l-file.

- m** Writes before each line of text in the g-file the SID of the delta that inserted the line into the SCCS file. The format is:
- SID tab line of text
- n** Writes the value of the %M% keyword before each line of text in the g-file (see "Identification Keywords" on page 480 for information on keywords). The format is the value of %M%, followed by a horizontal tab, followed by the text line. When both the **-m** and **-n** flags are used, the format is:
- %M% value tab SID tab line of text
- p** Writes the text created from the SCCS file to standard output and does not create a g-file. **get** sends output normally sent to standard output to file descriptor 2 instead. If you specify the **-s** flag with the **-p** flag, output normally sent to standard output does not appear anywhere. Do not use **-p** with the **-e** flag.
- rSID** Specifies the SCCS identification string (SID) of the SCCS file version to be created. Figure 2 on page 481 shows what version of a file is created and the SID of the pending delta as functions of the SID specified.
- s** Suppresses all output normally written to standard output. Error messages (written to standard error output), remain unaffected.
- t** Accesses the most recently created delta in a given release or release and level. Without the **-r** flag, **get** accesses the most recent delta regardless of its SID.
- wstring** Substitutes *string* for the %W% keyword in g-files not intended for editing (see "SCCS Files" on page 478 for information on g-files).
- xlist** Excludes a *list* of deltas in the creation of a file. See the **-i** flag for the SID list format on page 482.

Examples

1. To get an SCCS file for editing:

```
get -e s.prog.c
```

This creates a file named `prog.c` that only you have permission to modify. No one else can use `prog.c` or `s.prog.c` until you use the **delta** command to indicate that you are finished.

get

2. To get an SCCS file for reading:

```
get s.prog.c
```

This creates a file named `prog.c` that anyone can read, but that no one can modify. You can do this before searching files with the **grep** command or before compiling programs that are controlled with SCCS. If you are also using the **make** command to manage the development of a software project, **make** automatically does the **get** before compiling a program.

Related Information

The following commands: “**admin**” on page 41, “**delta**” on page 310, “**help**” on page 513, “**prs**” on page 781, and “**what**” on page 1213.

The **sccsfile** file in *AIX Operating System Technical Reference*.

The discussion of SCCS in *AIX Operating System Programming Tools and Interfaces*.

getopt

Purpose

Parses command line flags and parameters.

Syntax

```
set -- `getopt opstring $*`1
```

¹This command is not entered on the command line, but is used in shell procedures.

OL805050

Description

The **getopt** command is used to break up flags and parameters in command lines for easy parsing by shell procedures and to check for valid flags. *opstring* is a string of recognized flags (see the **getopt** subroutine call in *AIX Operating System Technical Reference*). Extended characters are not permitted. If a letter within *opstring* is followed by a colon, the flag is expected to take a modifying parameter that may or may not be separated from it on the command line by one or more tabs or spaces. If you specify -- as the last flag on a command line processed by **getopt**, **getopt** recognizes it and stops its processing; otherwise, **getopt** creates the terminating --. In either case, **getopt** places it at the end of the flags.

When the output from **getopt** is passed by command substitution to the shell **set** command, **set** resets all of the shell positional parameters (\$1, \$2 . . .) so that each flag is preceded by a - (minus) and occupies its own positional parameter. Each parameter (for example, file names and other parameters) is also parsed into its own positional parameter.

The **getopt** command writes a message to standard error when it encounters a flag not included in *opstring*. The **set** command returns a nonzero value if a flag appears on the command line but is not specified in *opstring*. Consequently, you can test the validity of command flags by testing the value of the shell variable \$? . If it is nonzero, the command line contains an unrecognized flag.

getopt

Example

The following shell procedure is a front end to the **ar** command. It uses **getopt** to separate the flags and parameters, then translates them into the **ar** command syntax and runs **ar** with these flags.

```
# @(#) lib: Front end to the ar command.
#
# Accepts the following flags:
#   ar flags with "-" prefixes. See the ar command.
#   -L library   The default library is "libsubs.a".
# Note: "lib -r -b sub1.o -v -l newsub.o" performs
#       "ar rrv1 sub1.o libsubs.a newsub.o".
#       The ar command DOES interpret this correctly.

set -- `getopt clsvmrua:b:i:dpqtxwL: $*`
if [ $? != 0 ]           # Test for syntax error
then
    exit 2
fi

FLAGS= POSNAME= LIBRARY=libsubs.a    # Default library name
while [ $1 != -- ]
do
    case $1 in
        -L)
            LIBRARY=$2
            shift; shift    # Shift past the -L and library name
            ;;
        -a|-b|-i)
            FLAGS=$FLAGS`expr "$1" : "-\(.\)\"`
            POSNAME=$2
            shift; shift    # Shift past the flag and parameter
            ;;
        -*)
            # Strip the "-" from the flag
            FLAGS=$FLAGS`expr "$1" : "-\(.\)\"`
            shift
            ;;
    esac
done

shift    # Shift past the "--" from getopt
FLAGS=${FLAGS:-vt}    # Default if action not specified
ar $FLAGS $POSNAME $LIBRARY $*
```


If this shell procedure is stored in a file named `lib`, then all of the following commands are equivalent:

```
lib -L mylib.a -v -r -b putfld.o getnam.o getfld.o getaddr.o
lib -Lmylib.a -v -r -bputfld.o getnam.o getfld.o getaddr.o
lib -Lmylib.a -vr bputfld.o getnam.o getfld.o getaddr.o
lib -Lmylib.a -v -rbputfld.o -- getnam.o getfld.o getaddr.o
```

In each of these cases, **getopt** breaks down the command into:

```
-L mylib.a -v -r -b putfld.o -- getnam.o getfld.o getaddr.o
```

The **getopt** command writes to its standard output. Because this command is enclosed in `` (grave accents), the shell takes its standard output and uses it to construct the command:

```
set -- -L mylib.a -v -r -b putfld.o -- getnam.o
getfld.o getaddr.o
```

This is called **command substitution**. For more details, see “Command Substitution” on page 925.

The **set** command (page 933) sets the positional parameters \$1, \$2, \$3 . . . to each of the values `-L`, `mylib.a`, `-v` . . . , respectively.

The shell procedure then uses the positional parameters to construct and run the command:

```
ar vrb putfld.o mylib.a getnam.o getfld.o getaddr.o
```

The **ar** command (page 55) accepts the flags in any order. Therefore, you can specify flags to `lib` in any order, as long as a parameter immediately follows a `-a`, `-b`, `-i`, or `-L` flag, and all the flags come before any file names. This means that:

```
lib -bputfld.o -rv -Lmylib.a getnam.o getfld.o getaddr.o
```

produces the command:

```
ar brv putfld.o mylib.a getnam.o getfld.o getaddr.o
```

which performs the same action as each of the previous commands. See “**test**” on page 1064 and “**expr**” on page 412 for more information about these commands.

Related Information

The following command: “**sh**” on page 913.

The **getopt** subroutine in *AIX Operating System Technical Reference*.

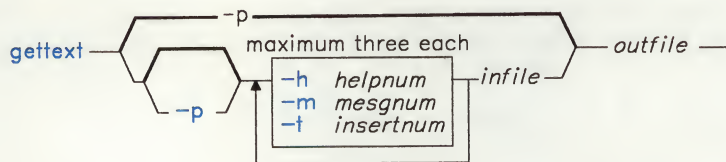
gettext

gettext

Purpose

Extracts message/insert/help descriptions.

Syntax



OL805130

Description

The **gettext** command gets message, insert, or help descriptions from *infile* and places the descriptions in *outfile*. If you specify the **-p** flag or **gettext outfile**, **gettext** places a message/insert/help template in *outfile*. When you have your message, insert, or help descriptions or your message/insert/help template in *outfile*; you can edit *outfile*.

The *outfile* is an AIX ASCII file that consists of a header to identify the component and a group of message/insert/help descriptions. The contents of the message/insert/help descriptions includes a delimiter, control information and message/insert/help text. See *AIX Operating System Programming Tools and Interfaces* for a description of the *outfile* format and contents.

Flags

- | | |
|---------------------|--|
| -h helpnum | Extracts help information from <i>infile</i> . You specify the index value used for the desired help number with <i>helpnum</i> . |
| -m mesgnum | Extracts message information from <i>infile</i> . You specify the index value used for the desired message number with <i>mesgnum</i> . |
| -p | Makes a message/insert/help template for <i>outfile</i> . |
| -t insertnum | Extracts text insert information from <i>infile</i> . You specify the index value used for the desired insert number with <i>insertnum</i> . |

The syntax for the *mesgnum*, *insertnum*, and *helpnum* parameters is as follows:

- num-num* Retrieves index numbers *num* to *num*.
- num,num . . .* Retrieves a list of index numbers specified with *num*, *num*, *num*, and so on (maximum of 50 numbers).
- num-* Retrieves index numbers equal to and larger than *num*.
- num* Retrieves index numbers from one to *num*.

Related Information

The following command: “**puttext**” on page 796.

The discussion of **gettext** in *AIX Operating System Programming Tools and Interfaces*.

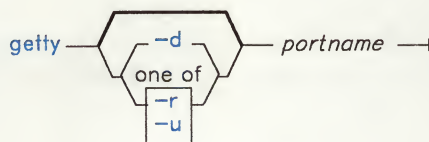
getty

getty

Purpose

Sets the characteristics of ports.

Syntax



OL805333

Description

The `init` process runs the `getty` command for each `portname` enabled for login. Its primary function is to set the characteristics of the port specified by `portname`. Port characteristics include:

- Bidirectional use (tty line can be used in both directions)
- Line speed (baud rate)
- Parity
- Carriage return, tab, new-line, and form feed delays
- Character set mapping, such as lowercase to uppercase, carriage return to new-line translation, and tab expansion
- Extended character support
- Character erase and line erase editing characters
- Local or remote echo
- Screen length for paging.

The `getty` command obtains these settings by reading the port attributes specified in the `/etc/ports` configuration file and by observing the behavior of the port itself. (For details regarding the format of `/etc/ports`, see *AIX Operating System Technical Reference*. For the `logmodes` and `runmodes` parameter settings, see “`stty`” on page 1018.)

When `getty` is invoked, it opens the specified port. However, if carrier detection (modem control) is available on the port, `getty` cannot open the port until the carrier is present. Once the port is opened, `getty` sets the work station attributes according to the first parameters in the `ports` file (for example, `speed`, `logmodes`, `parity`, `erase`, and `kill`). Then `getty` writes the message `herald` to the port and reads a login name from the port.

Note: If the login name contains extended characters, these extended characters are translated to the single ASCII characters that most resemble them.

Japanese Language Support Information

Login names are limited to ASCII characters.

If a framing error occurs while reading, either because a user generates a **BREAK** signal from the work station or because the line speed is not the same as that of the transmitting work station, the port parameters are reset to the next combination specified in the **ports** file.

Once **getty** reads a login name, it does the following:

1. Resets the work station modes according to the **runmodes** parameter.
2. Turns on carriage-return-to-new-line mapping if the login name was ended by a carriage return.
3. Turns on lowercase-to-uppercase mapping if the alphabetic characters in the login name were all uppercase.
4. Executes the program specified by the **logger** parameter. That program, defaulting to **/bin/login**, runs in the same process as **getty**, not as its child.

Any additional arguments entered after the login name are passed to the **logger** program. The **login** command interprets these as shell variable settings and places them in the environment.

On dial-in ports, it is often necessary to set no parity generation or checking as a default, but to permit the user to select parity as an option. For example, the following line in the **/etc/ports** file:

```
parity = none,odd+inpck,even+inpck
```

accepts logins with any parity. However, if a user generates **BREAK** before typing a login name, **getty** sets the port to generate odd parity and to check incoming characters for odd parity, while two **BREAKs** generate and check for even parity. Similarly, the following line:

```
speed=1200,300
```

works with 1200 baud, reverting to 300 baud when a **BREAK** is received before the login name. The default **runmodes** parameter (which must appear on one line in the **ports** file), is generally satisfactory. However, for work stations that have built-in tabs to every eight character positions and do not require tab delays, eliminating the **tab3** from the default in **/etc/ports** will provide faster output with less system load.

Notes for secure Command: After you run the **secure** command, the **/etc/ports** file can contain new attributes. These attributes are used by **getty** to manage the trusted path.

The following is a list of the possible attributes:

- sak** Turns on secure attention key (SAK) detection for the port and causes **getty** to perform a **frevoke** system call.
- synonym** This attribute specifies another port that requires the same protection and ownership as the port on which **getty** is working.
- shell** This attribute names the terminal manager program, usually **/bin/actman**. This program is passed to the **logger** program and is executed as the user's login shell when the user logs in successfully.

Special Purpose Options

If there is a **timeout** keyword in the **ports** file, **getty** waits only the specified number of seconds for a response to the herald before advancing to the next port settings or, after all the settings are exhausted, exiting. If there is a **program** keyword for the port, then instead of displaying the herald and gathering a login name, it executes the specified program immediately. This feature is a general mechanism for supporting special service ports such as network mail daemons that need to be spawned when a connection is made from the outside world. As a special case, if you specify:

program = HOLD

the **runmodes**, **owner**, and **protection** of the port are set and **getty** holds the port open indefinitely, thereby preventing the port modes from reverting to their open-default settings. This is useful, for example, in setting the modes on serial printer ports when it is inconvenient or impossible to have the programs that use them do so.

Flags

- d** Uses standard input as the work station for which parameters are to be set according to those governing *portname*. Instead of executing a logger or a program, **getty** displays the name of the program that would have been run.
- r** Makes the port available for shared (bidirectional) use. With this flag, **getty** attempts to create a lock file in **/etc/locks** with the name of the device. This file can then be used by **uucp** to determine the status of the line. If the lock fails (because some other process is using the line), **getty** waits until the lock file is removed, then exits. The **init** command creates a new **getty** to attempt the locking process again.
- u** Makes the port available for shared (bidirectional) use without displaying the login herald. This flag is used for direct lines or lines that have intelligent modems that need to return immediately on opening a port. This prevents **getty** from communicating with a **getty** on the remote system or modem.

Example

To test a new **/etc/ports** entry, enter the following:

```
getty -d /dev/tty5
```

This tests a new port definition for **/dev/tty5** by simulating the login sequence of this device at your work station.

Files

/bin/actman	A terminal manager program.
/etc/locks	Contains terminal devices to be locked
/etc/ports	Specifies terminal mode
/bin/login	Contains the authorization and identification program
/bin/setmaps	Contains characters for National Language Support (NLS)

Related Information

The following commands: “**login**” on page 584, “**init**” on page 521, and “**stty**” on page 1018.

The **tty** and **ports** files in *AIX Operating System Technical Reference*.

“Overview of International Character Support” in *Managing the AIX Operating System*.

The discussion of the trusted path in *Managing the AIX Operating System*.

The discussion of Japanese Language Support in *Japanese Language Support User's Guide*.

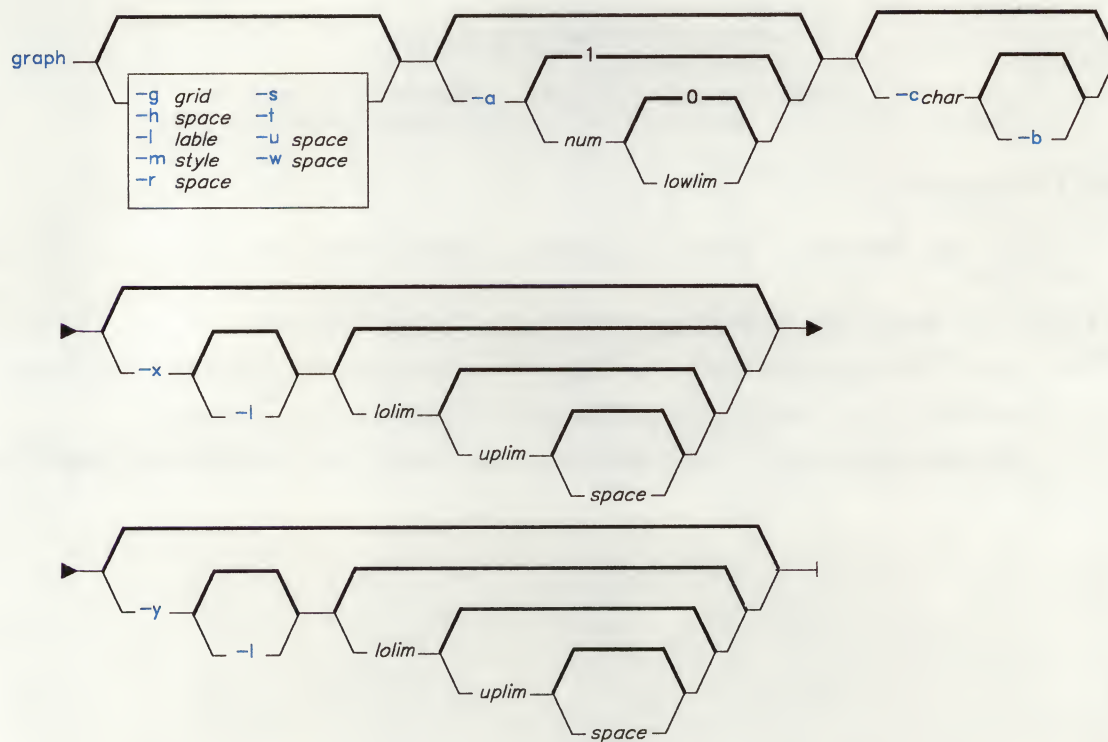
graph

graph

Purpose

Draws a graph.

Syntax



OL805429

Description

The **graph** command reads pairs of numbers from standard input, where each pair is the x and y coordinates of a point on a graph. It processes the data that allows the successive points to be connected by straight lines when printed and then writes the graph to standard output. See “**tplot**” on page 1079 for information on how to code the output for printing.

In the input, non-numeric strings following the coordinates of a point are labels. Labels begin on the point. Labels can be surrounded with " (double quotation marks), in which case they can be empty or contain blanks and numbers. Labels cannot contain new-line characters.

The **graph** command stores all points internally and drops those for which there is not room. It also drops segments that run out of bounds. The **graph** command produces a legend indicating grid range with a grid unless you specify the **-s** flag. If a specified lower limit exceeds the upper limit, **graph** reverses the axis. Note that logarithmic axes cannot be reversed.

Flags

- a** [*num* [*lolim*]]
Supplies abscissas missing from the input automatically. *num* determines the spacing on the axis (the default is 1). *lolim* determines the starting point for automatic abscissas (the default is 0 or the lower limit given by **-x**[*lolim*]).
- b**
Breaks the graph after each label in the input.
- c** *char*
Uses the character string *char* as the default label for each point.
- g** *grid*
Uses *grid* as the grid style, where *grid* = 0 indicates no grid, *grid* = 1 indicates a frame with tick marks, and *grid* = 2 indicates a full grid (default).
- h** *space*
Uses *space* as a fraction of space for height.
- l** "*label*"
Uses *label* as a label for the graph.
- m** *style*
Uses *style* as the style of connecting lines, where *style* = 0 indicates disconnected lines, and *style* = 1 indicates connected lines (default).
- r** *space*
Uses *space* as the fraction of space to move to the right before plotting.
- s**
Saves the current graphic screen image, does not erase before starting the plot.
- t**
Transposes horizontal and vertical axes. (**-x** now applies to the vertical axis).
- u** *space*
Uses *space* as the fraction of space to move up before plotting.
- w** *space*
Uses *space* as a fraction of space for width.

graph

-x [*l*] [*lolim* [*uplim* [*space*]]]

Makes the x axis logarithmic if *l* is used. Use *lolim* as the lower x axis limit and *uplim* as the upper x axis limit. Use *space* for the grid spacing on *x* axis. Normally these are determined automatically.

-y [*l*] [*lolim* [*uplim* [*space*]]]

Acts the same as **-x** for the y axis.

Related Information

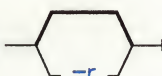
The following commands: “**spline**” on page 972 and “**tplot**” on page 1079.

graphics

Purpose

Accesses graphical and numerical commands.

Syntax

graphics 

OL777038

Description

The **graphics** command appends the path name **/usr/bin/graf** to the current **\$PATH** value, changes the primary shell prompt to **^**, and executes a new shell. The directory **/usr/bin/graf** contains all of the graphics subsystem commands.

The command line format for a command in **graphics** is *command name* followed by *argument(s)*. An *argument* may be a *filename* or an *flag string*. A *filename* is the name of any AIX Operating System file, except those beginning with **-**. The *filename -* is the name for the standard input. A *flag string* consists of **-** followed by one or more *flag(s)*. A *flag* consists of a key letter possibly followed by a value. Flags may be separated by commas.

The graphical commands have been partitioned into four groups.

- Commands that manipulate and plot numerical data; see “**stat**” on page 984.
- Commands that generate tables of contents; see “**toc**” on page 1074.
- Commands that interact with graphical devices; see “**gdev**” on page 460 and “**ged**” on page 463.
- A collection of graphical utility commands; see “**gutil**” on page 508.

To produce a list of **graphics** commands, enter **whatis** in the **graphics** environment.

Flag

- r** Creates access to the graphical commands in a restricted environment; that is, it sets **\$PATH** to **:/usr/bin/graf:/rbin:/usr/rbin** and invokes the restricted shell, **rsh**. To restore the environment that existed prior to issuing the **graphics** command, press **Ctrl-D** (END OF FILE). To log off of the graphics environment, enter **quit**.

Related Information

The following commands: “**gdev**” on page 460, “**ged**” on page 463, “**gend**” on page 475, “**gutil**” on page 508, “**stat**” on page 984, and “**toc**” on page 1074.

The **gps** file in *AIX Operating System Technical Reference*.

greek

Purpose

Converts output for a Teletype Model 37 work station to output for other work stations.

Syntax

```
greek -T$TERM -Tworkstation
```

OL805185

Description

The **greek** command reinterprets the Teletype Model 37 character set, including reverse and half-line motions, for display on other work stations. It simulates special characters, when possible, by overstriking. **greek** reads standard input and writes to standard output.

Flag

-Tworkstation	Uses the specified <i>workstation</i> . If you omit the -T flag, greek attempts to use the work station specified in the environment variable \$TERM (see the environ special facility in <i>AIX Operating System Technical Reference</i> .) <i>workstation</i> can be any one of the following:
300	DASI 300.
300-12	DASI 300 in 12-pitch.
300s	DASI 300s.
300s-12	DASI 300s in 12-pitch.
450	DASI 450.
450-12	DASI 450 in 12-pitch.
1620	Diablo 1620 (alias DASI 450).
1620-12	Diablo 1620 (alias DASI 450) in 12-pitch.
2621	Hewlett-Packard 2621, 2640, and 2645.
2640	Hewlett-Packard 2621, 2640, and 2645.
2645	Hewlett-Packard 2621, 2640, and 2645.
4014	Tektronix 4014.
hp	Hewlett-Packard 2621, 2640, and 2645.
tek	Tektronix 4014.

greek

Files

/usr/bin/300
/usr/bin/300s
/usr/bin/4014
/usr/bin/450
/usr/bin/hp

Related Information

The following commands: “**300**” on page 1262, “**4014**” on page 1264, “**450**” on page 1265, “**eqn, neqn, checkeq**” on page 395, “**hp**” on page 514, “**mm, checkmm**” on page 663, “**tplot**” on page 1079, and “**nroff, troff**” on page 709.

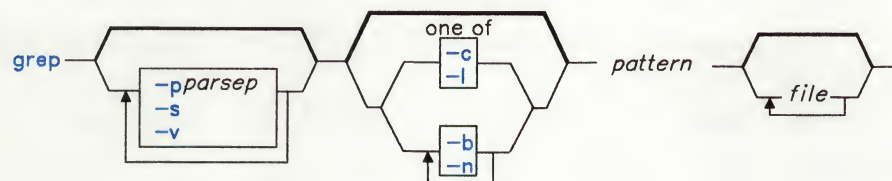
The **greek** miscellaneous facility in *AIX Operating System Technical Reference*.

grep

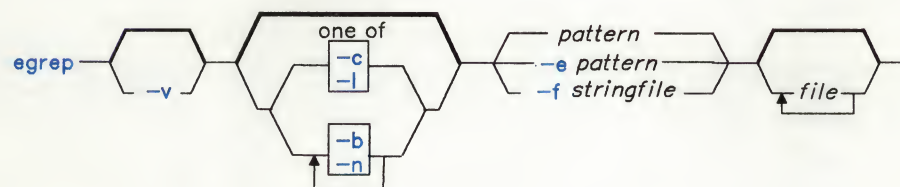
Purpose

Searches a file for a pattern.

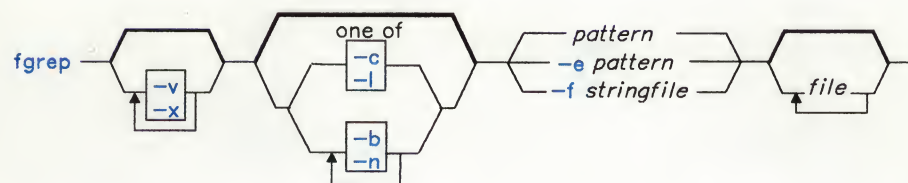
Syntax



OL805375



OL805359



OL805361

Description

Commands of the **grep** family search input *files* (standard input by default), for lines matching a pattern. Normally, they copy each line found to standard output. Three versions of the **grep** command permit you to express the matching pattern in varying levels of complexity:

grep Searches for *patterns*, which are limited regular expressions in the style of the **ed** command. **grep** uses a compact nondeterministic algorithm.

egrep Searches for *patterns* which are full regular expressions as in **ed**, except for \ (and \) and with the addition of the following rules:

- A regular expression followed by a plus sign (+) matches one or more occurrences of the regular expression.
- A regular expression followed by a question mark (?) matches 0 or 1 occurrences of the regular expression.
- Two regular expressions separated by a vertical bar (!) or by a new-line character match strings that are matched by either.
- A regular expression may be enclosed in parentheses () for grouping.

The order of precedence of operators is [], then * ? +, then concatenation, then ! and the new-line character.

The **egrep** command uses a deterministic algorithm that needs exponential space.

fgrep Searches for *patterns* that are fixed strings. It searches for lines that contain one of the strings (lines are separated by new-line characters).

All versions of **grep** display the name of the file containing the matched line if you specify more than one *file* name. Characters with special meaning to the shell (\$ * [! ^ () \), must be quoted when they appear in *patterns*. When *pattern* is not a simple string, you usually must enclose the entire *pattern* in single quotation marks. In an expression such as [a-z], the minus means “through” according to the current collating sequence.

A collating sequence may define **equivalence classes** for use in character ranges. See “Overview of International Character Support” in *Managing the AIX Operating System* for more information on collating sequences and equivalence classes.

Japanese Language Support Information

A collating sequence in Japanese Language Support does not define equivalence classes for use in character ranges. To avoid unpredictable results when using a range expression to match a class of characters, use a **character class expression** rather than a standard range expression. For information about character class expressions, see the discussion in “ed” on page 371.

The exit value of these commands is:

- 0 A match was found.
- 1 No match was found.
- 2 A syntax error was found or a file was inaccessible (even if matches were found).

Notes:

1. Lines are limited to 512 characters; longer lines are broken into multiple lines of 512 or fewer characters (**grep** only).
2. Paragraphs (under the **-p** flag) are currently limited to a length of 5000 characters.
3. Running **grep** on a special file produces unpredictable results and is discouraged.

Flags

- b** Precedes each line by the block number on which it was found. Use this flag to help find disk block numbers by context.
- c** Displays only a count of matching lines.
- e *pattern*** Specifies a *pattern*. This works the same as a simple *pattern* but is useful when the *pattern* begins with a - (does not work with **grep**).
- f *string file*** Specifies a file that contains patterns (**egrep**) or strings (**fgrep**).
- l** Lists just the names of files (once) with matching lines. Each file name is separated by a new-line character.
- n** Precedes each line with its relative line number in the file.
- pparsep** Displays the entire paragraph containing matched lines. Paragraphs are delimited by paragraph separators, *parsep*, which are patterns in the same form as the search pattern. Lines containing the paragraph separators are used only as separators; they are never included in the output. The default paragraph separator is a blank line (**grep** only).
- s** Suppresses error messages about inaccessible files (**grep** only).
- v** Displays all lines except those that match the specified pattern.
- x** Displays lines that match the pattern exactly with no additional characters (**fgrep** only).

Examples

1. To search several files for a simple string of characters:

```
fgrep "strcpy" *.c
```

e

This searches for the string `strcpy` in all files in the current directory with names ending in `.c`

2. To count the number of lines that match a pattern:

```
fgrep -c "{" pgm.c
fgrep -c "}" pgm.c
```

This displays the number of lines in `pgm.c` that contain open and close braces.

If you do not put more than one `{` or `}` on a line in your C programs, and if the braces are properly balanced, then the two numbers displayed will be the same. If the numbers are not the same, then you can display the lines that contain braces in the order that they occur in the file with: `egrep "{l}" pgm.c`

3. To use a pattern that contains some of the pattern-matching characters `*`, `^`, `?`, `[`, `]`, `\(`, `\)`, `\{`, and `\}`:

```
grep "^ [a-zA-Z]" pgm.s
```

This displays all lines in `pgm.s` that begin with a letter.

Note that because **fgrep** does not interpret pattern-matching characters:

```
fgrep "^ [a-zA-Z]" pgm.s
```

makes **fgrep** search only for the string `^ [a-zA-Z]` in `pgm.s`.

4. To use an extended pattern that contains some of the pattern-matching characters `+`, `?`, `|`, `(`, and `)`, ::

```
egrep "\( *[a-zA-Z]*|[0-9]*) *\)" my.txt
```

This displays lines that contain letters in parentheses or digits in parentheses, but not parenthesized letter-digit combinations. It matches `(y)` and `(783902)`, but not `(alpha19c)`.

Note:

When using **egrep**, `\(` and `\)` match parentheses in the text, but `(` and `)` are special characters that group parts of the pattern. The reverse is true for **grep**.

Japanese Language Support Information

When Japanese Language Support is installed on your system, use the following format:

```
egrep "\( *([[:alpha:]]*|[[:digit:]]*) *\)" my.txt
```

5. To display all lines that do *not* match a pattern:

```
grep -v "^#"
```

This displays all lines that do not begin with a # character.

6. To display the names of files that contain a pattern:

```
fgrep -l "strcpy" *.c
```

This searches the files in the current directory that end with .c and displays the names of those files that contain the string `strcpy`.

Related Information

The following commands: “**ed**” on page 371, “**sed**” on page 887, and “**sh**” on page 913.

“Overview of International Character Support” in *Managing the AIX Operating System*.

The discussion of Japanese Language Support in *Japanese Language Support User's Guide*.

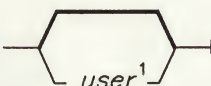
groups

groups

Purpose

Displays your group membership.

Syntax

`groups` 

¹The default *user* is the person running the command.

OL805129

Description

The **groups** command writes to standard output the groups to which you or the specified *user* belong. The AIX Operating System allows you to belong to many different groups at the same time.

Your **primary group** is specified in the file `/etc/passwd`. Once you are logged in, you can change your active group with the **newgrp** command (see page 689). When you create a file, its group ID is that of your active group.

Other groups that you belong to are specified in the file `/etc/group`. If you belong to more than one group, you can access files belonging to any of those groups without changing your primary group ID. These are called your **concurrent groups**.

Note: The `/etc/passwd` and `/etc/opasswd` files must be on the same node.

Japanese Language Support Information

This command has not been modified to support Japanese characters.

Files

<code>/etc/group</code>	Group file; contains group IDs.
<code>/etc/ogroup</code>	Previous version of the group file.
<code>/etc/passwd</code>	Password file; contains user IDs.
<code>/etc/opasswd</code>	Previous version of the password file.

Related Information

The following command: “**newgrp**” on page 689.

The **setgroups** system call and the **initgroups** subroutine in *AIX Operating System Technical Reference*.

gutil

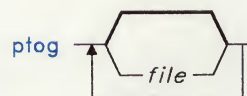
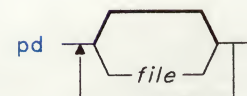
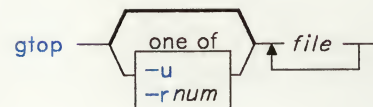
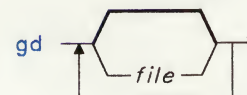
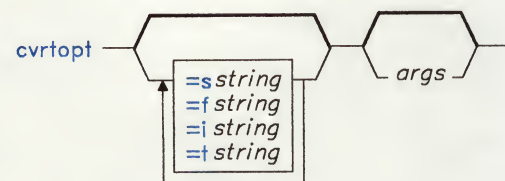
gutil

Purpose

Provides graphical utility programs.

Syntax

bel —



quit —

OL777039

OL805449

Flags

- cnum** Selects character set *num*, where *num* is an integer between 0 and 5. See the Hewlett-Packard 7221A Graphics Plotter documentation for a list of these character sets.
- pnum** Selects pen-numbered *num*, where *num* is an integer between 1 and 4 inclusive.
- rnum** Displays a window on a **GPS** region, where *num* is an integer from 1 to 25 inclusive.
- snum** Slants characters *num* degrees clockwise from the vertical.
- u** Displays window on the entire **GPS** universe.
- xdnum** Sets *x* displacement of the view port's lower left corner to *num* inches.
- xvnum** Sets width of view port to *num* inches.
- ydnum** Sets *y* displacement of the view port to *num* inches.
- yvnum** Sets height of view port to *num* inches.

erase

The **erase** command sends characters to a Tektronix 4010 series storage terminal to erase the screen.

hardcopy

When issued at a Tektronix display terminal with a hard copy unit, the **hardcopy** command produces a screen copy on the unit.

tekset

The **tekset** command send characters to a Tektronix terminal to clear the display screen, set the display mode to alpha, and set characters to the smallest font.

td

The **td** command translates a **GPS** object to scope code for a Tektronix 4010 series storage terminal. It computes a viewing window from the maximum and minimum points in *file*, unless you specify the **-u** or **-rnum** flag. Standard input is the default input file.

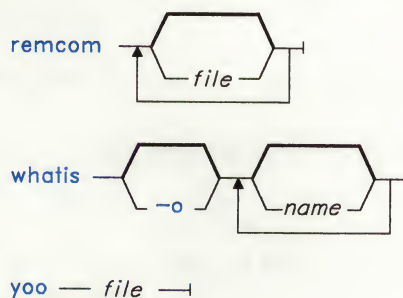
Flags

- e** Does not erase the screen before initiating display.
- rnum** Displays **GPS** region *num*, where *num* is an integer between 1 and 25 inclusive.
- u** Displays the entire **GPS** universe.

Related Information

The following commands: “**ged**” on page 463, “**gend**” on page 475, and “**graphics**” on page 497.

The **gps** file in *AIX Operating System Technical Reference*.



OL805450

Description

The following are the miscellaneous device-independent utility commands found in the `/usr/bin/graf` directory. If you do not specify any *files*, these commands read standard input. All output is sent to standard output. Graphical data is stored in **GPS** format; see the **gps** file in *AIX Operating System Technical Reference*.

bel

Sends the ASCII BEL character to the terminal.

cvrtopt

The **cvrtopt** command reformats its arguments (usually the command line arguments of a calling shell procedure) to facilitate processing by shell procedures. An *arg* is either a file name or a flag string. A file name is either a - (minus) by itself or a string not beginning with a -. A flag string is a string of flags beginning with a - (minus). **cvrtopt** produces output of the following form:

```
-flag -flag . . . file . . .
```

All flags appear singularly in the output and precede any file names. Arguments that take values or are two letters long must be described through flags to **cvrtopt**.

The **cvrtopt** command is usually used with the **set** command as the first line of a shell procedure (see page 933 for a description of the **set** command):

```
set - `cvrtopt [=flags] . . . $@`
```

Flags

sstring The specified *string* accepts string values, where *string* is a one or two letter flag name.

gutil

- fstring** The specified *string* accepts floating point numbers as values, where *string* is a one- or two-letter flag name.
- istring** The specified *string* accepts integers as values, where *string* is a one- or two-letter flag name.
- tstring** The specified *string* is a two-letter flag name that takes no value.

gd

The **gd** command produces a readable listing of a file in **GPS** format.

gtop

The **gtop** command transforms a **GPS** format into **plot** file commands displayable by **plot** filters. **GPS** objects are translated if they fall within the window that circumscribes the first *file*, unless you specify one of the following flags:

Flags

- rnum** Translates objects in **GPS** region *num*.
- u** Translates all objects in the **GPS** universe.

pd

The **pd** command displays a readable listing of **plot** format graphical commands.

ptog

The **ptog** command transforms **plot** file commands into a **GPS** file.

quit

The **quit** command terminates the session.

remcom

The **remcom** command copies its input to its output with comments removed. Comments are as defined in the C language (*/* comment */*).

whatis

The **whatis** command displays a short description of each *name* specified. If you do not specify a *name*, then **whatis** displays the current list of description *names*. The command **whatis *** displays every description.

Flag

-o Displays only command flags.

yoo

The **yoo** command is a piping primitive that deposits the output of a pipeline into a *file* used in the pipeline. Note that without **yoo**, this is not usually successful because it causes a read and write on the same file simultaneously.

Related Information

The following command: “**graphics**” on page 497.

The **gps** format in *AIX Operating System Technical Reference*.

hangman

hangman

Purpose

Plays hangman, the word-guessing game.

Syntax

`/usr/games/hangman` 

OL805228

Description

The **hangman** game chooses a word of at least seven letters from a standard dictionary. You try to guess the word by guessing the letters in it, one at a time. You are allowed seven mistakes. The *file* parameter specifies an alternate dictionary.

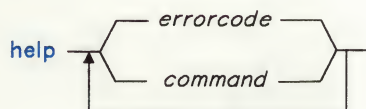
To quit the game, press INTERRUPT (Alt-Pause) or END OF FILE (Ctrl-D).

help

Purpose

Provides information about a Source Code Control System (SCCS) message or command or about certain non-SCCS commands.

Syntax



OL805054

Description

The **help** command writes to standard output information about the use of a specified SCCS *command* or about messages generated while using the commands. Each message has an associated *errorcode*, which can be supplied as a argument to the **help** command. Zero or more arguments may be supplied. If you do not supply a argument, **help** prompts for one. You may include any of the SCCS commands as arguments to **help**.

The *errorcode* consists of numbers and letters, and is found at the end of the message. For example, in the message `no id keywords (ge6)`, the error code is ge6.

Files

<code>/usr/lib/help</code>	Directory containing files of message text.
<code>/usr/lib/help/helploc</code>	File containing locations of help files not in <code>/usr/lib/help</code> .

Related Information

The discussion of SCCS in *AIX Operating System Programming Tools and Interfaces*.

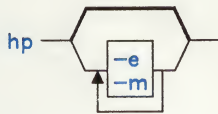
hp

hp

Purpose

Handles special functions for the HP2640- and HP2621-series terminals.

Syntax



OL805018

Description

The **hp** command reads standard input (usually output from **nroff**), and writes to standard output, which is usually Hewlett-Packard 2640-and 2621-series terminal displays.

If your terminal has the display enhancement feature, you can display subscripts and superscripts. With the mathematical-symbol feature, you can display Greek and other special characters the same way as the **300** command, with two exceptions: **hp** approximates the logical operator NOT with a right arrow and it only shows the top half of the integral sign.

For overstrike characters (characters followed by a backspace and another character), if either character is an underscore character, the other appears underlined or in inverse video depending on terminal enhancements.

Note: Some sequences of control characters (reverse line-feeds and backspaces) can make text disappear from the display. Tables with vertical lines generated by the **tbl** command will often be missing lines of text containing the bottom of a vertical line. You can avoid these problems by first piping the input through **col**, and then through **hp**.

Flags

- e Shows overstruck characters underlined, superscripts in half-bright, and subscripts half-bright underlined. Otherwise, all overstruck characters, subscripts, and superscripts appear in inverse video (dark-on-light). Use this flag only if your display has the display enhancements feature.
- m Produces only one blank line for any number of successive blank lines in the text.

Related Information

The following commands: “**300**” on page 1262, “**col**” on page 179, “**eqn, neqn, checkeq**” on page 395, “**greek**” on page 499, “**nroff, troff**” on page 709, and “**tbl**” on page 1053.

hyphen

hyphen

Purpose

Finds hyphenated words.

Syntax



OL805019

Description

The **hyphen** command reads the input *files* (standard input by default), finds all the lines ending with hyphenated words, and writes those words to standard output. A word is considered hyphenated only if the hyphen occurs at the end of a line. **hyphen** reads standard input if you do not specify any *file* names on the command line.

Note: The **hyphen** command cannot handle hyphenated italic words. It also sometimes gives unnecessary output.

Examples

1. To check the way words are hyphenated in a text file:

```
hyphen chap1
```

This lists the words in chap1 that are hyphenated at the end of a line.

2. To check the hyphenation performed by a text formatting program:

```
mm chap1 | hyphen
```

This lists the words that **nroff** decides to hyphenate across lines.

Related Information

The following commands: “**mm**, **checkmm**” on page 663, “**nroff**, **troff**” on page 709, and “**troff**” on page 710.

id

Purpose

Displays the system identity of the user issuing the command.

Syntax

`id` —

OL805131

Description

The **id** command writes a message to standard output containing the user and group IDs and corresponding names of the invoking process. When effective and real names and IDs do not match, **id** writes both.

Related Information

The following command: “**logname**” on page 589.

The **getuid** subroutine in *AIX Operating System Technical Reference*.

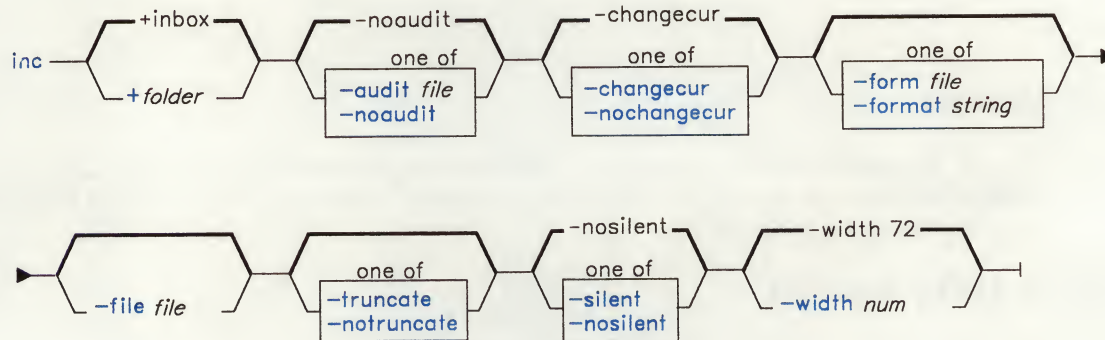
inc

inc

Purpose

Incorporates new mail.

Syntax



`inc -help`

AJ2FL156

Description

The **inc** command is used to incorporate incoming mail. **inc** is part of the MH (Message Handling) package and can be used with other MH and AIX commands.

The **inc** command takes all of the new messages from your mail drop and places them in the specified folder. If the specified folder does not exist, **inc** asks you if it should be created. **inc** assigns the new messages consecutive message numbers starting with the next highest number in the folder. **inc** assigns the new messages the protection code specified in the **Msg-Protect:** entry in your **.mh-profile**. If there is no **Msg-Protect:** entry, **inc** assigns the messages the protection code of 644. **inc** calls the **scan** command to display information about the new messages. If the **Unseen-Sequence:** profile entry specifies any sequences, **inc** adds the new messages to each sequence.

Flags

-audit <i>file</i>	Copies the current date to the specified file and appends the output from the scan command to the file.
-changeur	Sets the first new message as the current message for the specified folder. This flag is the default.
-file <i>file</i>	Incorporates messages from the specified file instead of the user's maildrop.
+folder	Incorporates the new messages into the specified folder. +inbox is the default folder.
-form <i>file</i>	Displays the scan command output in the alternate format described in <i>file</i> .
-format <i>string</i>	Displays the scan command output in the alternate format described by <i>string</i> .
-help	Displays help information for the command.
-noaudit	Does not record information about incorporating the new messages (see the -audit flag). This flag is the default.
-nochangeur	Does not alter the setting for the current message in the specified folder.
-nosilent	Prompts the user for any necessary information. This flag is the default.
-nottruncate	Does not clear the mailbox or file from which inc is taking new messages. If -file is specified, -nottruncate is the default.
-silent	Does not prompt you for any information. This flag is useful for running inc in the background.
-truncate	Clears the mailbox or file from which inc is taking new messages. If -file is not specified, -truncate is the default.
-width <i>num</i>	Sets the number of columns in the scan command output. The default is the width of the display.

Profile Entries

Alternate-Mailboxes:	Specifies your mailboxes.
Folder-Protect:	Sets the protection level for your new folder directories.
Msg-Protect:	Sets the protection level for your new message files.
Path:	Specifies your <i>user-mh-directory</i> .
Unseen-Sequence:	Specifies the sequences used to keep track of your unseen messages.

Files

<code>\$HOME/.mh-profile</code>	The MH user profile.
<code>/usr/lib/mh/mtstailor</code>	The MH tailor file.
<code>/usr/mail/\$USER</code>	The location of the mail drop.

Related Information

Other MH commands: “**mhmail**” on page 646, “**post**” on page 758, “**scan**” on page 871.

The **mh-format**, **mh-mail**, and **mh-profile** files in *AIX Operating System Technical Reference*.

“Overview of the Message Handling Package” in *Managing the AIX Operating System*.

init

Purpose

Initializes the system.

Syntax

`init` —¹

¹This command should not be entered on the command line.

OL805132

Description

After the kernel completes the basic processor initialization, it starts the **init** process, which is the ancestor of all other processes in the system. The **init** command controls the mode in which the system is running; this usually is either maintenance mode or multiuser mode. A user with superuser authority can alter **init** so that the system runs in controlled access mode. It is the program from which all loggers and most system daemons start.

When **init** starts, it determines what the startup mode should be, based on information in the `/etc/.init.state` file. If this file does not exist or is unreadable, **init** bases the startup mode on information passed to it by the kernel. The following are the usual startup modes for **init**:

maintenance	Starts a shell on the console, but does not start any other processes (single-user mode).
multiuser	Runs the command file <code>/etc/rc</code> and spawns loggers on all enabled ports.
exec-program	Runs the specified program.

Maintenance Mode

Use maintenance mode for the following:

- System installation
- Correcting problems on the file system using the **fsck** command
- All operations requiring an inactive system.

There are three ways to bring the system up in maintenance mode:

1. If the system is currently running in normal (multiuser) mode, use the **shutdown -m** command to bring the system down to maintenance mode (**shutdown** sends **init** a **SIGINT** signal).
2. Start the system from the Installation/Maintenance Diskette and specify the Maintenance Mode option from the End System Management menu.
3. Edit the **/etc/.init.state** file so that it consists of the character **m**. This causes the system to come up in maintenance mode each time it is started up.

Maintenance mode starts a shell program with superuser authority on the console. When you log off this shell by pressing END OF FILE (**Ctrl-D**), **init** asks you if you want to leave maintenance mode. A response beginning with **n** or **N** indicates "no," and **init** starts another shell on the console. Any processes running in the background continue to run. Any other response indicates "yes."

If the response is yes, **init** enters normal mode, as described in the following section. It also asks if the file system integrity should be checked. A response beginning with **n** or **N** indicates "no." Any other response indicates "yes." Your answer determines whether the **rc** command runs with an **m** or **d** argument.

Normal Mode

After the normal startup of the system (either from system startup or by leaving maintenance mode), **init** runs the normal initialization command file, **/etc/rc**. It passes **rc** an argument of either **m** (normal startup, clean root), or **d** (normal startup, dirty root). The latter is the default argument if the startup is from maintenance mode. **rc** is responsible for performing integrity checks, doing any necessary cleanups, mounting the normal file systems, enabling standard ports, and starting system daemons. If an error occurs during the running of this command file (indicated by a nonzero return code), **init** either forces a system restart by executing the **reboot** system call or enters maintenance mode.

Once **rc** completes successfully, **init** starts logger processes (normally **getty**) on each enabled port. Whenever someone ends a logger by logging off a port, **init** notes the logoff and starts a new logger on the port. Everything **init** knows about enabling ports is contained in the **/etc/portstatus** file, which is maintained by the **penable** command. Through this file, you can enable new ports or disable ports that were previously enabled. Whenever **init** receives a **SIGHUP** (hangup) signal, it rereads the **portstatus** file to see if any changes of port status have been requested.

The **init** commands then reads the commands in the **/etc/rc.ds** file, if that file exists. Typically, **/etc/rc.ds** contains commands to start Distributed Services. Any commands that are needed to run remote mounts should be placed in **/etc/rc.ds**.

If, at any time after the system starts up normally, **init** discovers that no ports are enabled or if **init** receives an **INTERRUPT** signal, it decides again on startup options. Generally, this means **init** will go through normal startup, assuming a dirty root.

Environments

Because **init** is the ultimate ancestor of every process on the system, its environment parameters are inherited by every process. As part of its initialization sequence, **init** reads the file **/etc/environment** and copies any assignments found in that file into the environment passed to all of its subprocesses. It treats **umask** differently. If it is assigned a reasonable octal value, **init** does a **umask** system call for the specified value, rather than passing the value in the environment. Similarly, if **filesize** is specified, **init** issues a **ulimit** call with the given size as the argument.

Files

/etc/utmp	Record of logged-in users
/usr/adm/wtmp	Permanent login accounting file
/etc/portstatus	Enabled port status file
/etc/rc	Initialization command file
/etc/environment	System environment variables

Related Information

The following commands: “**getty**” on page 490, “**pstart**, **penable**, **pshare**, **pdelay**” on page 791, “**rc**” on page 806, and “**shutdown**” on page 946.

The **reboot** and **umask** system calls and the **portstatus** file in *AIX Operating System Technical Reference*.

The discussion of the trusted path in *Managing the AIX Operating System* and *Using the AIX Operating System*.

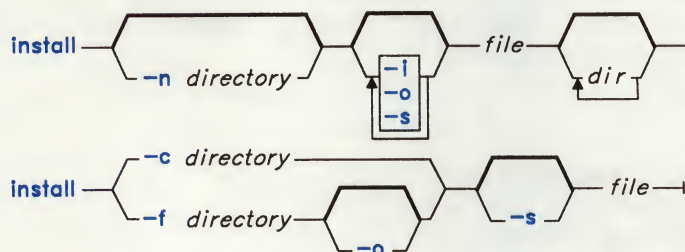
install

install

Purpose

Installs a command.

Syntax



OL805022

Description

The **install** command installs *file* in a specific place within a file system. It is most often used in makefiles (see “**make**” on page 625). When replacing files, **install** copies each file into the appropriate directory, thereby retaining the original owner and permissions. A newly-created *file* has permission code 755, owner **bin**, and group **bin**. **install** writes a message telling you exactly which files it is replacing or creating and where they are going.

If you do not supply any arguments, **install** searches a set of default directories (**/bin**, **/usr/bin**, **/etc**, **/lib**, and **/usr/lib**, in that order) for a file with the same name as *file*. The first time it finds one, it overwrites it with *file* and issues a message indicating that it has done so. If a match is not found, **install** issues a message telling you there was no match and exits with no further action.

If any directories are specified on the command line, **install** searches them before it searches the default directories.

Japanese Language Support Information

This command has not been modified to support Japanese characters.

Flags

- c** *directory* Installs a new command file in *directory* only if that file does not already exist there. If it finds a copy of *file* there, it issues a message and exits without overwriting the file. This flag can be used alone or with **-s**.
- f** *directory* Forces installation of *file* in *directory* whether or not *file* already exists. If the file being installed does not already exist, **install** sets the permission code and owner of the new file to **755** and **bin**, respectively. This flag can be used alone or with **-o** or **-s**.
- i** Ignores the default directory list and searches only those directories specified on the command line. This flag cannot be used with **-c** or **-f**.
- n** *directory* Installs *file* in *directory* if it is not in any of the searched directories and sets the permissions and owner of the file to **755** and **bin**, respectively. This flag cannot be used with **-c** or **-f**.
- o** Saves the old copy of *file* by copying it to **OLDfile** in the directory in which it found it. This flag cannot be used with **-c**.
- s** Suppresses display of all but error messages.

Examples

1. To replace a command that already exists in one of the default directories:

```
install fixit
```

This replaces *fixit* if it is found in **/bin**, **/usr/bin**, **/etc**, **/lib**, or **/usr/lib**. Otherwise, it is not installed. For example, if **/usr/bin/fixit** exists, then this file is replaced by a copy of the file *fixit* in the current directory.

2. To replace a command that already exists in a specified or default directory, and to preserve the old version:

```
install -o fixit /etc /usr/games
```

This replaces *fixit* if found in **/etc**, **/usr/games**, or one of the default directories. Otherwise it is not installed. If *fixit* is replaced, the old version is preserved by renaming it **OLDfixit** in the directory in which it was found (**-o**).

3. To replace a command that already exists in a specified directory:

```
install -i fixit /u/jim/bin /u/joan/bin /usr/games
```

This replaces *fixit* if found in **/u/jim/bin**, **/u/joan/bin**, or **/usr/games**. Otherwise it is not installed.

install

4. To replace a command if found in a default directory, or install it in a specified directory if not found:

```
install -n /usr/bin fixit
```

This replaces `fixit` if found in one of the default directories. If `fixit` is not found, it is installed as `/usr/bin/fixit (-n /usr/bin)`.

5. To install a new command:

```
install -c /usr/bin fixit
```

This creates a new command by installing a copy of `fixit` as `/usr/bin/fixit`, but only if this file does not already exist.

6. To install a command in a specified directory whether or not it already exists:

```
install -f /usr/bin -o -s fixit
```

This forces `fixit` to be installed as `/usr/bin/fixit` whether or not it already exists. The old version, if any, is preserved by moving it to `/usr/bin/OLDfixit (-o)`. The messages that tell where the new command was installed are suppressed (`-s`).

Related Information

The following command: “**make**” on page 625.


The **mk** system maintenance procedure in *AIX Operating System Technical Reference*.

install-mh

Purpose

Initializes the MH environment.

Syntax

`install-mh`  `-auto` `|`

AJ2FL230

Description

The **install-mh** command is used to set up mailbox directories. **install-mh** is not designed to be run directly by the user; it is designed to be called by other programs. The **install-mh** command is part of the MH (Message Handling) package.

The **install-mh** command is issued automatically the first time you run any MH command. **install-mh** prompts you for the name of your mail directory. If the directory does not exist, **install-mh** asks you if it should be created. **install-mh** creates the file **\$HOME/.mh-profile** and places the **Path:** profile entry in it. This entry identifies the location of your mailbox.

Flag

-auto Creates the standard MH path without prompting.

Profile Entry

Path: Specifies your *user-mh-directory*. This entry is created by **install-mh**.

install-mh

Files

\$HOME/.mh-profile The MH user profile.

Related Information

The **mh-profile** file in *AIX Operating System Technical Reference*.

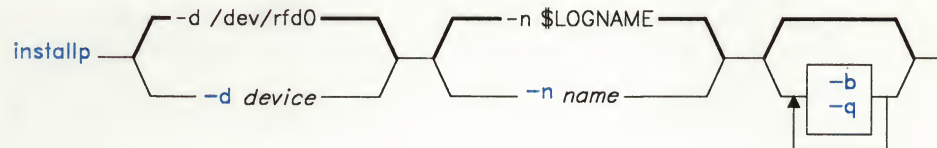
“Overview of the Message Handling Package” in *Managing the AIX Operating System*.

installp

Purpose

Installs a licensed program.

Syntax



OL805021

Description

Warning: Before you install a program, restart your system and make sure that no other users are on the system and no other programs are running.

The **installp** command installs a program. You must be a member of the system group or operating with superuser authority to run this command.

When auditing is on, an audit record of the type, **installp** is created. If your system is running in controlled access mode, this command should only be run under the **watch** command.

Because more than one program may be on a set of diskettes, **installp** asks whether or not you want to install each program. If you do, **installp** checks to see if it is an older version than the one currently installed. If the version to be installed is older than the version on the system, **installp** informs you and asks if you want to continue.

When **installp** is instructed to perform the installation of a program subset for code service clients on a system where the complete program is already installed, it checks to see if the program subset is compatible with the existing complete program. If the program subset and complete program are incompatible, **installp** warns you and then prompts you for a response.

The **installp** command makes a backup copy of the program history file before installation begins. If installation is unsuccessful, it sets the Version, Release, and Level fields of the last record of the history file to 00.00.0000 and logs the exit value in the program history file. The history file remains on the system as `/usr/lpp/pgm-name/lpp.hist`, where *pgm-name* is the program name.

installp

Pressing INTERRUPT (Alt-Pause) will not stop the **installp** command. To stop **installp**, press QUIT WITH DUMP (Ctrl-V). Use this only in extreme circumstances since the system state is unpredictable and one of the following can occur:

- The write-verify feature may be left on for all minidisks
- All terminals other than the console may be disabled
- Some install control files may need to be deleted.

Notes:

1. Only ordinary files with the prefix **lpp.** remain in **/usr/lpp/pgm-name** after completion of **installp**. All other ordinary files are removed.
2. Installing a complete program on a client where a program subset already exists causes the history file information for the program subset to be destroyed.

Flags

- b Runs the **bffcreate** command to create an installation file in backup format from the distribution media. Then tells **installp** to use the file as the distribution media for the install. A server in a code service environment uses this flag to install.
- d *device* Installs the program from the specified *device*. The default *device* is **/dev/rfd0**.
- n *name* Logs the first eight nonblank characters of *name* in the program history file. The default *name* is the value of the environment variable **\$LOGNAME**.
- q Runs in quiet mode suppressing most of the interactive queries.

The **installp** command runs a program-provided installation procedure **instal**. Each installation procedure returns one of the following exit values to **installp**:

- 0 Installation completed. Take no action.
- 2 Update superblocks, i-nodes, and delayed block I/O (sync), then restart the AIX Operating System.
- 3 Build the kernel, then update the superblocks, i-nodes, and delayed block I/O (sync) and shut down the AIX Operating System.
- 4 Build the kernel, then update the superblocks, i-nodes, and delayed block I/O (sync) and restart the kernel.
- 5 Installation cancelled without errors.
- 6 Update superblocks, i-nodes, and delayed block I/O (sync), then shut down the AIX Operating System.

Any other return value indicates that installation failed.

Internal Commands

Install procedures can use the internal install commands. These commands provide common code for the save and recovery functions frequently needed by most program-provided procedures. They do a minimum validation of input parameters and return exit values like subcommands. You can, however, receive messages from system commands that they call. C Language procedures that call these commands can use the `/usr/include/inu21.h` file to define return codes.

inuse

The **inuse** command saves some or all of the files and archive files that will change during a program install or update procedure. It uses the following syntax:

inuse *listfile* *pgm-name*

The *pgm-name* parameter specifies the program to be installed or updated. *pgm-name* can be a maximum of eight characters. *listfile*, which must be a full path name, contains a list of relative path names (relative to the root) for all of the files that need to be saved. *listfile* must be in the format of an **apply list**. (See *AIX Operating System Programming Tools and Interfaces* for a discussion of the format of an apply list.)

The **inuse** command creates the **save directory** (`/usr/lpp/pgm-name/inst_updt.save`). This is the directory in which the install and update procedures store saved files and the control list that correlates the local file names with their full path names. **inuse** uses *listfile* as a basis to determine which files need to be temporarily saved.

If the file named in *listfile* already exists, **inuse** copies that file to `/usr/lpp/pgm-name/inst_updt.save/update.n`, where *n* is an integer assigned by **inuse**. If the file does not exist, **inuse** assumes that this entry in *listfile* represents either a new file or a file to be archived or processed by the archive procedure. **inuse** maintains a list of saved files in `/usr/lpp/pgm-name/inst_updt.save/update.list`. The format of each entry in the list is:

update.n *file*

where **update.n** is the name of the saved file and *file* is the full path name of the file.

An archived constituent file is saved if there is a valid archive control file, `/usr/lpp/pgm-name/lpp.acf`, for the program. If this file exists, **inuse** compares each of the file names in *listfile* to the constituent file names in `/usr/lpp/pgm-name/lpp.acf`. When it finds a match, **inuse** uses the **ar** command to extract the constituent file from its associated archive file.

installp

It then moves it to `/usr/lpp/pgm-name/inst_updt.save/archive.n`, where *n* is an integer selected by **inuse**. **inuse** maintains a list of the extracted files that have been saved in the file `/usr/lpp/pgm-name/inst_updt.save/archive.list`. The format of each entry in the list is:

archive.n *cfile* *afile*

where **archive.n** is the name of the saved file and *cfile* and *afile* are the constituent and archive files defined in the archive control file.

The **inuse** command returns the following exit values:

- 0 No error conditions occurred.
- 105 Failure occurred trying to create a save directory.
- 107 Copy of a file from one directory to another failed. This implies that the update apply has not yet begun and that the old level of the program is still usable.
- 202 One or more parameters missing.
- 204 Too many parameters were entered.
- 207 Could not access the apply list.

inurecv

The **inurecv** command recovers all files and archive-constituent files saved from the previous **inuse**. **inurecv** uses the following syntax:

inurecv *pgm-name* *reject-flag*

It uses the control lists from the `/usr/lpp/pgm-name/inst_updt.save` directory to recover the files. **inuse** creates the `/usr/lpp/pgm-name/inst_updt.save` directory and control lists. **inurecv** also recovers files that may have been saved by the program-provided install or update procedure (see *AIX Operating System Programming Tools and Interfaces* for details).

The **inurecv** command has to distinguish between an immediate recovery that occurs because of an error condition during an install or update and an update rejection that occurs because a user rejects an update (`updatep -r`). If the *reject-flag* argument is **yes**, **inurecv** assumes that it is being run because of an update rejection. If the argument is **no** or if no flag is specified, **inurecv** assumes that it is being run because of an immediate recovery.

The **inurecv** command returns the following exit status values:

- 0 No error conditions occurred.
- 101 The save directory does not exist.
- 102 A copy of a file from one directory to another failed. This implies that the program could not be recovered and that it must be reinstalled and any updates reapplied.
- 104 A file that was saved in the save directory was not found.
- 205 Replacement of a constituent file in an archive file failed while attempting to recover a program. This implies that the program is no longer useable and should be reinstalled and any updates reapplied.

inurest

The **inurest** command does simple restores and archives. It does not do any additional processing or user interaction. **inurest** uses the following syntax:

```
inurest [-ddevice] [-q] listfile pgm-name
```

The *listfile* is the full path name of a file containing the relative directory target path name (relative to the root), of files that a program needs to restore. It must be in the format of an apply list. **inurest** restores all files in the list relative to the root directory. *pgm-name* specifies the name of the program to be installed or updated. It can be a maximum of eight characters.

To archive a file, there must be an archive control file, */usr/lpp/pgm-name/lpp.acf*. If it exists, **inurest** compares each of the target names in *listfile* to the component files listed in there. Whenever **inurest** finds a match, it archives the restored file into the corresponding archive file and deletes the restored file.

Flags

The following flags modify the action of **inurest**:

- d device** Specifies the input *device*. The default device is */dev/rfd0*.
- q** Prohibits **restore** from displaying the insert volume 1 prompt.

The **inurest** command returns the following exit status values:

- 0 No error conditions occurred.
- 106 Failed trying to restore an updated version of files.
- 201 An invalid flag was specified.
- 202 One or more parameters missing.
- 204 Too many parameters were entered.
- 206 Failed trying to replace file in an archive file.
- 208 Could not access the apply list.

ckprereq

The **ckprereq** command determines whether the system level is compatible with the program to be installed or updated. It uses the following syntax:

```
ckprereq [-v] [-fprerequisites]
```

You can run **ckprereq** only if you are a member of the system group or are operating with superuser authority. *prerequisites* is a program prerequisite list file. Each record in this file contains the name of a prerequisite program and describes the version, release, and level requirements. There is one record for each prerequisite program. The default *prerequisites* file is **prereq**. See *AIX Operating System Programming Tools and Interfaces* for details on the format of **ckprereq** file entries.

installp

The **ckprereq** command tests the current version, release, and level found in the history file and marks each **prereq state** field of the **prereq** file with one of the following codes if the test fails:

l	The test is false for level.
f	The history file format is not fixed 80.
n	The history file was not found.
r	The test is false for release.
s	There is a syntax error in the prereq file.
u	The history file is in an unknown state.
v	The test is false for version.

A blank **prereq state** field indicates that the test was true. The exit value of **ckprereq** is the number of records that did not test true. If all records test true, the exit value is 0.

Note: If a program is installed on a local node and executed on a remote node, the remote node must have file trees that have all necessary prerequisite files available. When auditing is on, an audit record of the type, **ckprereq** is created.

Flags

-f prerequisites	Specifies the <i>prerequisites</i> file to use in place of prereq .
-v	Sends a descriptive message to standard error for each failure in the prerequisite program test. The messages give the same information as the prereq state field of the prereq file.

mvmd

The **mvmd** command updates the VRM minidisk. It uses the following syntax:

```
mvmd -a file -D VRM-dir [-fp [file]] -l pgm-name  
mvmd -c VRM-file permissions -l pgm-name  
mvmd -d VRM-file -l pgm-name  
mvmd -m VRM-file [-fp [file]] -l pgm-name  
mvmd -r file -D VRM-dir -l pgm-name
```

You must be a member of the system group or operating with superuser authority to run **mvmd**. When auditing is on, an audit record of the type, **mvmd** is created.

Flags

-a file	Adds the specified <i>file</i> to the VRM minidisk. Use the -D flag to specify the destination VRM directory. <i>file</i> must not already exist in the specified directory. By default, mvmd adds the file to the first unused position in the VRM directory. To specify a position, use the -f or -p flag.
----------------	--

-c VRM-file permissions

Changes the permission code of the specified *VRM-file* to the octal value, *permissions*. The *VRM-file* parameter must be a full path name. Valid combinations of permission bits are as follows:

- 0700** The loadlist processor loads, runs, and deletes this module.
- 0450** The loadlist processor transfers control to this module after all loadlist directory entries have been processed.
- 0440** The loadlist processor loads this module.
- 0410** This module is a virtual machine.
- 0040** If the system startup device is a diskette, the loadlist processor is to load the module. If the system startup device is a fixed disk, the loadlist processor does not load the module. Instead, it maps the module.

The loadlist processor ignores any module that does not have the load bit set. For more information about these permission bits, see *Virtual Resource Manager Technical Reference*.

- d VRM-file** Deletes the specified file from the VRM minidisk. The *VRM-file* parameter must be a full path name.
- D VRM-dir** Specifies the full path name of the VRM directory.
- f [file]** Specifies the position following *file* in the directory list or, if you do not specify *file*, the bottom of the directory list. Use this positioning flag with the **-a** or **-m** flags.
- l pgm-name** Specifies the name of a program that is modifying the VRM minidisk. The *pgm-name*, the date, the user name, and a descriptive title are placed in a record appended to the VRM history file. If you do not specify this flag, then a record with the name UNKNOWN is appended to the VRM history file.
- m VRM-file** Moves the specified file within its VRM directory. By default, **mvmd** moves the file to the first unused position. To specify a position, use the **-f** or **-p** flag.
- p [file]** Specifies the position prior to *file* in the directory list or, if you do not specify *file*, the top of the directory list. Use this positioning flag with the **-a** or **-m** flags.
- r file** Replaces the specified *file* on the VRM minidisk. Use the **-D** flag to select the VRM directory of the file to be replaced. Both the replacement file and the file to be replaced must have the same name.

The **mvmd** command returns an exit status of 0 if no errors occurred. A nonzero return indicates that an error occurred.

installp

Files

/usr/include/inu21.h	Defines error codes for internal commands
/usr/lpp/ <i>pgm-name</i> /instal	Program installation procedure
/usr/lpp/ <i>pgm-name</i> /inst_updt.save	Directory for saved files
/usr/lpp/ <i>pgm-name</i> /inst_updt/inuPIDtempn	Temporary files
/usr/lpp/ <i>pgm-name</i> /liblpp.a	Central archive file
/usr/lpp/ <i>pgm-name</i> /lpp.acf	Archive control file
/usr/lpp/ <i>pgm-name</i> /lpp.hist	Program history file
/usr/lpp/ <i>pgm-name</i> /prereq	Program prerequisite list file
/usr/lpp/ <i>pgm-name</i> /inst_updt.save/archiven	File containing saved, archived, constituent files
usr/lpp/ <i>pgm-name</i> /inst_updt.save/archive.list	List of the saved, extracted, archived files

Related Information

The following commands: “**updatep**” on page 1122 and “**bffcreate**” on page 108.

The discussion of code service in *Managing the AIX Operating System*.

The **fork** and **exec** system calls and the **lpp.hist** file in *AIX Operating System Technical Reference*.

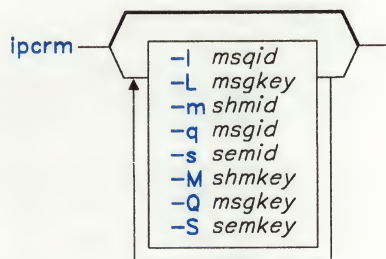
The discussion of installing programs in *AIX Operating System Programming Tools and Interfaces*.

ipcrm

Purpose

Removes message queue, semaphore set or shared memory identifiers.

Syntax



OL805135

Description

The **ipcrm** command removes one or more message queue, semaphore set, or shared memory identifiers.

Japanese Language Support Information

This command has not been modified to support Japanese characters.

Flags

- | | |
|-------------------------|---|
| -l <i>msgid</i> | Removes local information about the remote queue <i>msgid</i> without removing the remote queue. |
| -L <i>msgkey</i> | Removes local information about the remote queue <i>msgkey</i> without removing the remote queue. |
| -m <i>shmid</i> | Removes the shared memory identifier <i>shmid</i> . The shared memory segment and data structure associated with <i>shmid</i> are also removed after the last detach. |

ipcrm

- M *shmkey*** Removes the shared memory identifier, created with key *shmkey*. The shared memory segment and data structure associated with it are also removed after the last detach.
- q *msqid*** Removes the message queue identifier *msqid* and the message queue and data structure associated with it.
- Q *msgkey*** Removes the message queue identifier, created with key *msgkey*, and the message queue and data structure associated with it.
- s *semid*** Removes the semaphore identifier *semid* and the set of semaphores and data structure associated with it.
- S *semkey*** Removes the semaphore identifier, created with key *semkey*, and the set of semaphores and data structure associated with it.

The details of the remove operations are described in **msgctl**, **shmctl**, and **semctl** in the *AIX Operating System Technical Reference*. The identifiers and keys can be found by using the **ipcs** command.

Related Information

The following command: “**ipcs**” on page 539.

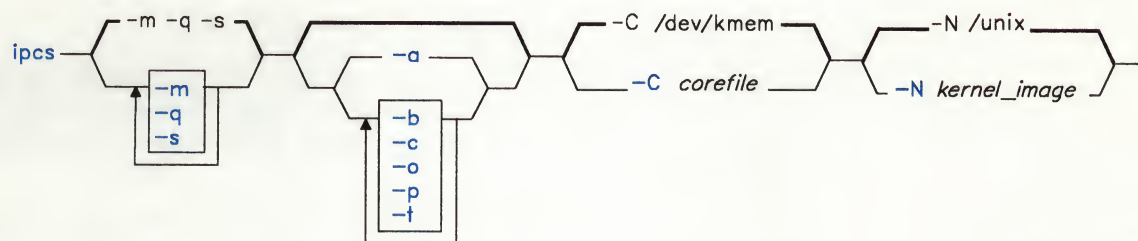
The **msgctl**, **msgget**, **msgrcv**, **msgsnd**, **semctl**, **semget**, **semop**, **shmctl**, **shmget**, and **shmop** system calls in *AIX Operating System Technical Reference*.

ipcs

Purpose

Reports interprocess communication facility status.

Syntax



OL805432

Description

The **ipcs** command writes to the standard output information about active interprocess communication facilities. If you do not specify any flags, **ipcs** writes information in a short form about currently active message queues, shared memory segments, semaphores, remote queues, and local queue headers.

The column headings and the meaning of the columns in an **ipcs** listing follow. The letters in parentheses indicate the flags that cause the corresponding heading to appear. **all** means that the heading always appears. These flags only determine what information is provided for each facility. They do not determine which facilities will be listed.

T (all) Type of facility:

- q** message queue
- Q** message queue resides on a remote node
- m** shared memory segment
- s** semaphore.

ID (all) The identifier for the facility entry.

KEY (all) The key used as a parameter to **msgget**, **semget**, or **shmemget** to make the facility entry.

Note: The key of a shared memory segment is changed to **IPC_PRIVATE** when the segment is removed until all processes attached to the segment detach it.

MODE (all) The facility access modes and flags. The mode consists of 11 characters that are interpreted as follows:

The first two characters can be:

- R** if a process is waiting on a **msgrcv**
- S** if a process is waiting on a **msgsnd**
- D** if the associated shared memory segment has been removed. It disappears when the last process attached to the segment detaches it.
- C** if the associated shared memory segment is to be cleared when the first attach is run
- if the corresponding special flag is not set.

The next 9 characters are interpreted as three sets of 3 bits each. The first set refers to the owner's permissions; the next to permissions of others in the user-group of the facility entry; and the last to all others. Within each set, the first character indicates permission to read, the second character indicates permission to write or alter the facility entry, and the last character is currently unused.

The permissions are indicated as follows:

- r** if read permission is granted
- w** if write permission is granted
- a** if alter permission is granted
- if the indicated permission is *not* granted.

OWNER (all) The login name of the owner of the facility entry.

GROUP (all) The name of the group that owns the facility entry.

CREATOR (a,c) The login name of the creator of the facility entry.

CGROUP (a,c) The group name of the group of the creator of the facility entry.

Note: For the **OWNER**, **GROUP**, **CREATOR**, and **CGROUP**, the user and group IDs display instead of the login names.

CBYTES (a,o) The number of bytes in messages currently outstanding on the associated message queue.

QNUM (a,o) The number of messages currently outstanding on the associated message queue.

QBYTES (a,b) The maximum number of bytes allowed in messages outstanding on the associated message queue.

LSPID (a,p) The ID of the last process that sent a message to the associated queue. If the last message sent was from a process in a node other than the node which holds the queue, then **LSPID** is the PID of the kernel process which actually placed the message on the queue, not the PID of the sending process.

LRPID	(a,p) The ID of the last process that received a message from the associated queue. If the last message received was from a process in a node other than the node which holds the queue, then LRPID is the PID of the kernel process which actually received the message on the queue, not the PID of the receiving process.
STIME	(a,t) The time when the last message was sent to the associated queue. For remote queues, this is the server time. No attempt is made to compensate for time-zone differences between the local clock and the server clock.
RTIME	(a,t) The time when the last message was received from the associated queue. For remote queues, this is the server time. No attempt is made to compensate for any clock skew between the local clock and the server clock.
CTIME	(a,t) The time when the associated entry was created or changed. For remote queues, this is the server time. No attempt is made to compensate for any clock skew between the local clock and the server clock.
NATTCH	(a,o) The number of processes attached to the associated shared memory segment.
SEGSZ	(a,b) The size of the associated shared memory segment.
CPID	(a,p) The process ID of the creator of the shared memory entry.
LPID	(a,p) The process ID of the last process to attach or detach the shared memory segment.
ATIME	(a,t) The time when the last attach was completed to the associated shared memory segment.
DTIME	(a,t) The time the last detach was completed on the associated shared memory segment.
NSEMS	(a,b) The number of semaphores in the set associated with the semaphore entry.
OTIME	(a,t) The time the last semaphore operation was completed on the set associated with the semaphore entry.

Japanese Language Support Information

This command has not been modified to support Japanese characters.

Flags

- a** Uses the **-b**, **-c**, **-o**, **-p** and **-t** flags.
- b** Writes the maximum number of bytes in messages on queue for message queues, the size of segments for shared memory, and the number of semaphores in each semaphores set.
- c** Writes the login name and group name of the user that made the facility.
- Ccorefile** Uses the file *corefile* in place of **/dev/kmem**. *corefile* is a memory image file produced by the **Ctrl-(left)Alt-End** key sequence.
- m** Writes information about active shared memory segments.
- Nkernel-image** Uses the specified *kernel-image* (**/unix** is the default).
- o** Writes the following usage information:
 - Number of messages on queue
 - Total number of bytes in messages in queue for message queues
 - Number of processes attached to shared memory segments.
- p** Writes the following:
 - Process number of the last process to receive a message on message queues
 - Process number of the creating process
 - Process number of last process to attach or detach on shared memory segments.
- q** Writes information about active message queues.
- s** Writes information about active semaphore set.
- t** Writes the following:
 - Time of the last control operation that changed the access permissions for all facilities
 - Time of the last **msgsnd** and last **msgrcv** on message queues
 - Time of the last **shmat** and last **shmdt** on shared memory
 - Time of the last **semop** on semaphore sets.

Files

/unix	System kernel image
/dev/kmem	Memory
/etc/passwd	User names
/etc/group	Group names

Related Information

The **ipcs**, **msgrev**, **msgsnd**, **semop**, **shmat**, and **shmdt** system calls in *AIX Operating System Technical Reference*.

The discussion of generating core files in *Problem Determination Guide*.

ipctable

ipctable

Purpose

Creates, displays, or changes the Distributed Services IPC Queues Table.

Syntax

`ipctable` —

OL805468

Description

The **ipctable** command lets you build, examine, or modify the Distributed Services IPC Queues Table. Only members of the system group or users operating with superuser authority can use **ipctable** to change the state of the Distributed Services IPC Queues Tables (see “su” on page 1026). Other user can use **ipctable** to browse the IPC Queues Table.

Related Information

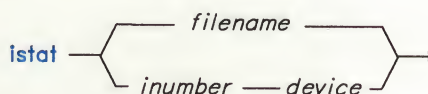
“Getting Started With Distributed Services Configuration Menus” in *Managing the AIX Operating System*.

istat

Purpose

Examines i-nodes.

Syntax



OL805138

Description

The **istat** command writes information about the i-nodes specified with *inumber* to standard output. Use the **istat** command to write information about the i-node for a specified *filename*, or to write the contents of a specified i-node, *inumber* on an arbitrary file system.

If you specify *filename*, **istat** writes the following information about the file:

- The device where the file resides.
- The i-node number of the file, on that device.
- The file type (normal, directory, block device, and so on).
- What protection is on the file.
- The name and identification number of the owner and group.

Note: The owner and group names for remote files are taken from the local */etc/passwd* file.

- The number of links to the file.
- If the i-node is for a normal file, the length of the file.
- If i-node is for a device, the major and minor device designations.
- The date of the last time the i-node was updated.
- The date of the last time the file was modified.
- The date of the last time the file was referenced.

If you specify *inumber* and *device*, **istat** also displays, in long decimal values, the block numbers recorded in the i-node. You can specify the *device* as either a device name or as a mounted-file-system name.

Note: *inumber* and *device* cannot specify a remote device.

Japanese Language Support Information

This command has not been modified to support Japanese characters.

Examples

1. To display the information stored in a file i-node:

```
istat /bin/sh
```

This displays the i-node information for the file /bin/sh. The information looks something like this:

```
Inode 34 on device 0/10 File
Protection: rwxr-xr-x Sticky
Owner: 0(su) Group: 0(system)
Link count: 1 Length 54240 bytes
```

```
Last updated: Tue Dec 18 01:07:36 1984
Last modified: Sat Jun 30 18:11:47 1984
Last accessed: Wed Feb 13 11:06:37 1985
```

2. To display i-node information if given a file i-number:

```
istat 34 /dev/hd0
```

This displays the information contained in i-node number 34 on the /dev/hd0 device. In addition to the information shown in Example 1, this displays:

```
Block pointers:
    219  220  221  222  223  224  225  226
    227  228  229   0   0
```

These numbers are addresses of the disk blocks that contain the data in the file.

Related Information

The following command: “**fsdb**” on page 450.

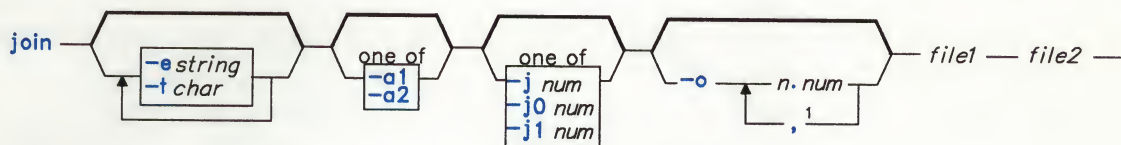
The **stat** system call and the **filesystems** and **fs** files in *AIX Operating System Technical Reference*.

join

Purpose

Joins data fields of two files.

Syntax



¹ Do not put a blank on either side of the comma.

OL805371

Description

The **join** command reads *file1* and *file2*, joins lines in the files according to the flags, and writes the results to standard output. Both files must be sorted according to the collating sequence specified by the **NLCTAB** environment variable, if set, for the fields on which they are to be joined (normally the first field in each line).

One line appears in the output for each identical **join field** appearing in both *file1* and *file2*. The join field is the field in the input files that **join** looks at to determine what will be included in the output. The output line consists of the join field, the rest of the line from *file1*, then the rest of the line from *file2*. You can specify standard input in place of *file1* by substituting a - (minus) for the name.

Both input files must be sorted in increasing ASCII collating sequence on the fields on which they are to be joined (the join field, normally the first field in each line).

Fields are normally separated by a blank, a tab character, or a new-line character. In this case, **join** treats consecutive separators as one, and discards leading separators.

join

Flags

- anum** When *num* is **1**, **join** produces an output line for each line found in *file1* but not in *file2*. When *num* is **2**, **join** produces an output line for each line found in the *file2* but not in *file1*.
- e string** Replaces empty output fields with *string*.
- j[n] num** Joins the two files on the *numth* field of file *n*. *n* is **0** or **1**. If you do not specify *n*, **join** uses the *numth* field in each file.
- o n.num[,n.num . . .]** Makes each output line consist of the fields specified in *list*, in which each element has the form *n.num*, where *n* is a file number and *num* is a field number.
- tchar** Uses *char* as the field separator character in the input and the output. Every appearance of *char* in a line is significant. The default separator is a blank. With default field separation, the collating sequence is that of **sort -b**. If you specify **-t**, the sequence is that of a plain sort. To specify a tab character, enclose it in single quotation marks (").

Examples

Note: The vertical alignment shown in these examples may not be consistent with your output.

1. To perform a simple join operation on two files whose first fields are the same:

`join phonedir names`

If phonedir contains the following telephone directory:	and names contains these names and departments numbers:	then join displays:
Adams A. 555-6235 Dickerson B. 555-1842 Erwin G. 555-1234 Jackson J. 555-0256 Lewis B. 555-3237 Norwood M. 555-5341 Smartt D. 555-1540 Wright M. 555-1234 Xandy G. 555-5015	Erwin Dept. 389 Frost Dept. 217 Nicholson Dept. 311 Norwood Dept. 454 Wright Dept. 520 Xandy Dept. 999	Erwin G. 555-1234 Dept. 389 Norwood M. 555-5341 Dept. 454 Wright M. 555-1234 Dept. 520 Xandy G. 555-5015 Dept. 999

Each line consists of the join field (the last name), followed by the rest of the line found in `phonedir` and the rest of the line in `names`.

2. To display unmatched lines with the command:

```
join -a2 phonedir names
```

If phonedir contains:	and names contains:	then join displays:
Adams A. 555-6235	Erwin Dept. 389	Erwin G. 555-1234 Dept. 389
Dickerson B. 555-1842	Frost Dept. 217	Frost Dept. 217
Erwin G. 555-1234	Nicholson Dept. 311	Nicholson Dept. 311
Jackson J. 555-0256	Norwood Dept. 454	Norwood M. 555-5341 Dept. 454
Lewis B. 555-3237	Wright Dept. 520	Wright M. 555-1234 Dept. 520
Norwood M. 555-5341	Xandy Dept. 999	Xandy G. 555-5015 Dept. 999
Smartt D. 555-1540		
Wright M. 555-1234		
Xandy G. 555-5015		

This performs the same join operation as in Example 1, and also lists the lines of names that have no match in phonedir. Frost and Nicholson are included in the listing, since they do not have entries in phonedir.

3. To display selected fields:

```
join -o 2.3 2.1 1.2 1.3 phonedir names
```

This displays the following fields in the order given:

Field 3 of names	(Department Number)
Field 1 of names	(Last Name)
Field 2 of phonedir	(First Initial)
Field 3 of phonedir	(Telephone Number)

If phonedir contains:	and names contains:	then join displays:
Adams A. 555-6235	Erwin Dept. 389	389 Erwin G. 555-1234
Dickerson B. 555-1842	Frost Dept. 217	454 Norwood M. 555-5341
Erwin G. 555-1234	Nicholson Dept. 311	520 Wright M. 555-1234
Jackson J. 555-0256	Norwood Dept. 454	999 Xandy G. 555-5015
Lewis B. 555-3237	Wright Dept. 520	
Norwood M. 555-5341	Xandy Dept. 999	
Smartt D. 555-1540		
Wright M. 555-1234		
Xandy G. 555-5015		

join

4. To perform the join operation on a field other than the first:

```
sort +2 -3 phonedir | join -j1 3 - numbers
```

This combines the lines in `phonedir` and `numbers`, comparing the third field of `phonedir` to the first field of `numbers`.

First, this sorts `phonedir` by the third field, because both files must be sorted by their join fields. The output of `sort` is then piped to `join`. The `-` (minus sign) by itself causes the `join` command to use this output as its first file. The `-j1 3` defines the third field of the sorted `phonedir` as the join field. This is compared to the first field of `numbers` because its join field is not specified with a `-j` flag.

If numbers contains:	then this command displays the names listed in <code>phonedir</code> for each telephone number:
555-0256	555-0256 Jackson J.
555-1234	555-1234 Erwin G.
555-5555	555-1234 Wright M.
555-7358	

Note that `join` lists all the matches for a given field. In this case, `join` lists both Erwin G. and Wright M. as having the telephone number 555-1234. The number 555-5555 is not listed because it does not appear in `phonedir`.

Related Information

The following commands: “`awk`” on page 81, “`comm`” on page 183, “`sort`” on page 958, “`cut`” on page 269, and “`paste`” on page 736.

The **environment** miscellaneous facility in *AIX Operating System Technical Reference*.

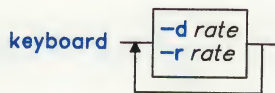
“Overview of International Character Support” in *Managing the AIX Operating System*.

keyboard

Purpose

Controls the delay and repetition rates of the keyboard.

Syntax



OL805443

Description

The **keyboard** command changes the keyboard delay and repetition rates. These rates are initially set at system startup to 500 milliseconds and 14 characters per second, respectively.

Flags

- d rate** Sets the delay rate to the specified value. *rate* can be 300, 400, 500, or 600 milliseconds.
- rrate** Sets the rate of repetition to the specified value. This *rate* can be an integer from 2 to 40, inclusive.

Example

To change both the delay and repetition rates:

```
keyboard -d300 -r40
```

This sets the keyboard to a delay of 300 milliseconds and the repetition rate to 40 characters per second.

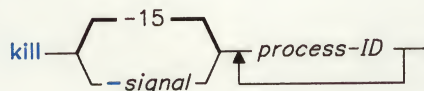
kill

kill

Purpose

Sends a signal to a running process.

Syntax



OL805139

Description

The **kill** command sends a *signal* to a running process, by default signal 15 (**SOFTWARE TERMINATE**). This default action normally kills processes that do not catch or ignore the signal. You specify a process by giving its *process-ID* (process identification number, or **PID**). The shell reports the PID of each process that is running in the background (unless you start more than one process in a pipeline, in which case the shell reports the number of the last process). You can also use the **ps** command to find the process ID number of commands.

In addition, there are special *process-IDs* that cause the following special actions:

- 0** The *signal* is sent to all processes having a process-group ID equal to the process-group ID of the sender (except those with PID's 0 and 1).
- 1** If the effective user ID of the sender is not 0 (root), *signal* is sent to all processes with a process-group ID equal to the effective user ID of the sender. (except those with PID's 0 and 1).
If the effective user ID of the sender is 0 (root), *signal* is sent to all processes, excluding numbers 0 and 1.
- process-ID** The *signal* is sent to all processes whose process-group number is equal to the absolute value of *process-ID*. Note that when you specify a minus PID, you must also specify the *signal* to be sent, even signal 15.

See the **kill** system call in *AIX Operating System Technical Reference* for a complete discussion of **kill**. For a list of signal numbers, see the **signal** systems call in *AIX Operating System Technical Reference*.

Unless you are operating with superuser authority, the process you wish to stop must belong to you. When operating with superuser authority, you can stop any process.

Examples

1. To stop a given process:

```
kill 1095
```

This stops process 1095 by sending it the default signal, 15 (also called **SIGTERM**). Note that process 1095 might not actually stop if it has made special arrangements to ignore or override signal 15.

2. To stop several processes that ignore the default signal:

```
kill -9 1034 1095
```

This sends signal 9 (**SIGKILL**) to processes 1034 and 1095. Signal 9 is a special signal that normally cannot be ignored or overridden.

3. To stop all of your background processes:

```
kill 0
```

This sends signal 15 to all members of the shell process group. This includes all background processes started with **&**. (See page 914 about running background processes.) Although the signal is sent to the shell, it has no effect because the shell ignores signal 15.

4. To stop all of your processes and log yourself off:

```
kill -9 0
```

This sends signal 9 to all members of the shell process group. Because the shell cannot ignore signal 9, this also stops the login shell and logs you off. If you are using multiple windows on a high-function terminal, then this closes the active window.

5. To stop all processes that you own:

```
kill -9 -1
```

This sends signal 9 to all processes owned by the effective user, even those started at other work stations and that belong to other process groups. If you are using multiple windows on a high-function terminal, then this closes all of the windows. If a listing that you requested is being printed, then it is also stopped.

Note: To send signal 15 with this form of the **kill** command, you must specify **-15** explicitly:

```
kill -15 -1
```


kill

6. To send a different signal code to a process:

```
kill -16 1103
```

This sends signal 16 (**SIGUSR1**) to process 1103.

The name of the **kill** command is misleading because many signals, including 16, do not stop processes. The action taken on signal 16 is defined by the particular application you are running.

Related Information

The following commands: “**cs**h” on page 225, “**ps**” on page 786, and “**sh**” on page 913.

Note: The **cs**h command contains a built-in subcommand named **kill**. The command and subcommand do not necessarily work the same way. For information on the subcommand, see the **cs**h command.

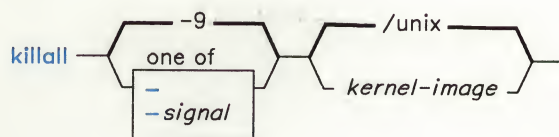
The **kill** and **signal** system calls in *AIX Operating System Technical Reference*.

killall

Purpose

Cancels all processes except the calling process.

Syntax



OL805140

Description

The **killall** command cancels all processes that you started, except those producing the **killall** process. This command provides a convenient means of canceling all processes created by the shell that you control. When started by a user operating with superuser authority, **killall** cancels all cancelable processes except those that started it.

The *kernel-image* parameter specifies the name of the system load module (by default, **/unix**).

Flags

- Sends a **SIGTERM** signal initially and then sends a **SIGKILL** (kill) signal to all processes that survive for 30 seconds after receipt of the signal first sent. This gives processes that catch **SIGTERM** signal an opportunity to clean up. (For more information, see the **signal** system call in *AIX Operating System Technical Reference*.)
- signal* Sends the specified *signal* number. (For information about signal numbers, see the **signal** system call in *AIX Operating System Technical Reference*.)

Examples

1. To stop all background processes that have started:

```
killall
```

This sends all background processes the kill signal 9 (also called **SIGKILL**).

killall

2. To stop all background processes, giving them a chance to clean up:

```
killall -
```

This sends signal 15 (**SIGTERM**), waits 30 seconds, and then sends signal 9 (**SIGKILL**).

3. To send a specific signal to the background processes:

```
killall -2
```

This sends signal 2 (**SIGINT**) to the background processes.

Files

/unix	System kernel image.
/dev/mem	Used for reading the process table.

Related Information

The following command: “**kill**” on page 552.

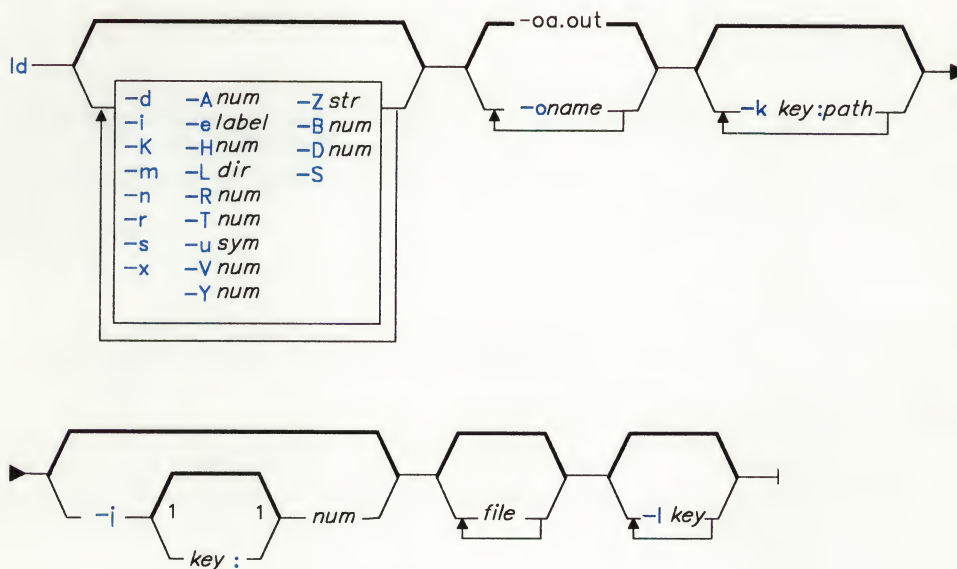
The **signal** system call in *AIX Operating System Technical Reference*.

ld

Purpose

Links object files.

Syntax



¹ Do not put a blank between these items.

OL805362

OL805308

Description

The `ld` command (the *linkage editor*) combines the specified object *files* into one file, resolving external references and searching libraries. It produces an object module that can be run or that can become a *file* parameter in another call to `ld`. In the latter case, you must use the `-r` flag to preserve the relocation bits. `ld` places its output in a file named `a.out`. It makes this file executable if no errors occur during the link and if the `-r` flag is not specified.

The linkage editor links object files and searches object libraries in the order specified. It links object modules unconditionally, but links from the library only those files that define an unresolved external reference. If a routine from a library calls another routine in that library, the called routine must follow the calling routine.

Unless you use the **-e** flag to specify another entry point, the first byte of the first nonnull text segment (or the first byte of the data segment if all text segments are null) becomes the entry point of the output file.

The reserved symbols **_text**, **_data**, **_sdata**, **_etext**, **_edata**, and **_end** (in C, **text**, **data**, **sdata**, **etext**, **edata**, and **end**) are set to the first location of the program, the first location of the data, the segment number of the data, the first location above the program, the first location above initialized data, and the first location above all data, respectively. You cannot define these symbols.

Because you can use **ld** to link modules intended to run on other machines, some of its action depends upon the architecture of the computer system on which you intend to run the module. **ld** recognizes that architecture automatically from the input modules and modifies its action accordingly. You can use some of its flags to alter the default behavior of **ld** for a particular architecture.

Flags

The **ld** command recognizes several flags. Except for **-l** entries, which are really abbreviations for file names, the order in which you specify flags does not affect the way they work. You can specify numeric values in either decimal, octal (with a leading **0**), or hexadecimal (with a leading **0x** or **0X**) format.

- Anum** Stores *num* in the **a-misc** field of the output file header. This field indicates the size of memory, in bytes, allocated to the process which runs the file. On many systems, the stand-alone loader or kernel uses this value to set the base of the run-time stack pointer.
- Bnum** Makes *num* the starting address for the uninitialized data (bss) segment of the output file. The default starting address is the first storage unit after the end of the data segment. Not all architectures support the separation of data and bss segments.
- d** Defines common storage, even if you have specified the **-r** flag.
- Dnum** Makes *num* the starting address for the initialized data segment of the output file. The default starting address begins at location 0 (if **-i** is in effect), at the first storage unit after the end of the text segment, or, if **-n** is in effect, at the next page or segment boundary.
- elabel** Makes *label* the entry point of the executable output file.
- Hnum** Makes *num* the boundary, usually the page size, to which the text segment must be padded if it has a different protection than does the data segment. Specify this parameter only to override the default value for the given architecture.

- i** Assigns text and data segments to separate address spaces in memory, with the text segment read-only—if the architecture supports read-only memory—and shared among all users. The data segment starts at location zero unless set with **-D**. If the architecture does not support separate instruction and data space, this flag is treated as if it were **-n**.
- j[key:]num** Assigns the shared library image *key* to location *num*. If you do not specify *key*, do not use location *num* when you assign the run-time location of the shared library text images. The exact interpretation of *num* depends on the target architecture. On the RT work station, *num* refers to the segment register, one of 4 through 13. You can specify **-j** once for each shared library image that has an assigned location.
- kkey:path** Maps any reference to the shared library image with the shared library *key* into *path*. Instead of adding the shared library *key* to the run-time table, add *path*. You can specify **-k** once for each shared library image with a remapped *key*.
- K** Loads the **a.out** header into the first bytes of the text segment, followed by the text segments from the object modules. This flag causes pages of executable files to be aligned on pages in the file system so that they can be paged upon demand on systems that support paging. This flag provides mapped file support for the text and data segments.
- lkey** Searches the specified library file, where *key* selects the file **libkey.a**. **ld** searches for this file in the directory specified by an **-L** flag, then in **/lib** and **/usr/lib**. It searches library files in the order that you list them on the command line.
- Ldir** Looks in *dir* for files specified by **-l** keys. If it does not find the file in *dir*, **ld** searches the standard directories.
- m** Lists on standard output the names of all files and archive members used to create the output file.
- n** Makes the text segment read-only—if the architecture supports read-only memory—and shared among all users running the file. The data segment starts at the first segment boundary following the end of the text unless set with **-D**. On architectures which only permit read-only text with separate text and data spaces, the **-n** flag is treated as if it were the **-i** flag.
- o name** Assigns *name* rather than **a.out** to the output file.
- r** Writes relocation bits in the output file so that it can serve as a file parameter in another **ld** call. This flag also prevents common symbols from being assigned final definitions and suppresses the undefined symbol diagnostic messages.
- Rnum** Makes *num* bytes the allocation unit for objects manipulated by **ld**, such as segments or common objects. Typically this value ranges from 1 to 8. Specify this parameter only to override the default value for the given architecture.

- s** Strips the symbol table, line number information, and relocation information from the output. This saves space but impairs the usefulness of the debugger. Using the **strip** command has the same effect. This flag is turned off if there are any undefined symbols.
- S num** Makes *num* the maximum size the user stack is allowed to grow. This value represents the number of bytes allowed. If you do not specify this argument, the system assumes a default limit of 1 MB.
- Tnum** Makes *num* the starting address for the text segment of the output file. If not specified, the text segment begins at location zero.
- u sym** Enters *sym* into the symbol table as an undefined symbol. This is useful when linking from only a library, since initially the symbol table is empty and an unresolved reference is needed to force the linking of the first routine.
- Vnum** Stores *num* in the **a-version** field of the output file header; *num* must be in the range 0 to 32767.
- x** Does not enter local symbols in the output symbol table; enters only external symbols. This flag saves some space in the output file.
- Ynum** In a segmented system, makes *num* the boundary to which the text segment should be padded if it has a protection different from that of the data segment. If *num* is zero, the padding is either that selected by the **-H** flag or the default value associated with that flag. Specify this parameter only to override the default value for the given architecture.
- Zstr** Prefixes with *str* the names specified by the **-l** key. For example, with **-Z/test** and **-lxyz**, **ld** looks for the file **/test/lib/libxyz.a** or, if that file does not exist, **/test/usr/lib/libxyz.a**. The ordinary directories will not be searched. This flag is most useful when cross-compiling.

Examples

1. To link several object files and produce an **a.out** file to run under the AIX Operating System without the Floating-Point Accelerator:

```
ld -n -t0x10000000 -K /lib/crt0.o pgm.o subs1.o subs2.o -lrts -lc
```

A simpler way to accomplish this is to use the **cc** command to link the files as follows:

```
cc pgm.o subs1.o subs2.o
```

Since the **cc** command automatically uses the link options and necessary support libraries, you do not need to specify them on the command line (it gets this information from the configuration file **cc.cfg**). For this reason, you should use **cc** to link files when you are producing programs that run under the AIX Operating System.

2. To specify the name of the output file:

```
cc -o pgm pgm.o subs1.o subs2.o
```

This stores the linked output in the file `pgm`.

3. To conditionally link library subroutines:

```
cc pgm.o subs1.o subs2.o mylib.a -ltools
```

This links the object modules `pgm.o`, `subs1.o`, and `subs2.o` unconditionally. It then links the subroutines from `mylib.a` that are used by the preceding modules. (This is often called **conditional linking**.) Then **ld** conditionally links subroutines from the library specified by `-ltools`. (This means `/lib/libtools.a`, if it exists. If **ld** does not find this file, then it looks for `/usr/lib/libtools.a`.)

Note: Always list libraries and `-l` flags at the end of the **ld** or **cc** command lines.

Files

<code>/lib/lib*.a</code>	Libraries.
<code>/usr/lib/lib*.a</code>	Libraries.
<code>a.out</code>	Output file.

Related Information

The following commands: “**ar**” on page 55, “**as**” on page 61, “**cc**” on page 140, and “**shlib**” on page 939.

The **a.out** file in *AIX Operating System Technical Reference*.

The discussion of **ld** in *AIX Operating System Programming Tools and Interfaces* and in *Assembler Language Reference*.

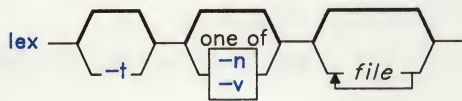
lex

lex

Purpose

Generates a C Language program that matches patterns for simple lexical analysis of an input stream.

Syntax



OL805025

Description

The **lex** command reads *file* or standard input, generates a C Language program, and writes it to a file named **lex.yy.c**. This file, **lex.yy.c**, is a compilable C Language program.

The **lex** command uses *rules* and *actions* contained in *file* to generate a program, **lex.yy.c**, which can be compiled with the **cc** command. It can then receive input, break the input into the logical pieces defined by the *rules* in *file*, and run program fragments contained in the *actions* in *file*. For a more detailed discussion of **lex** and its operation, see *AIX Operating System Programming Tools and Interfaces*.

The generated program is a C Language function called **yylex**. **lex** stores **yylex** in a file named **lex.yy.c**. You can use **yylex** alone to recognize simple, one-word input, or you can use it with other C Language programs to perform more difficult input analysis functions. For example, you can use **lex** to generate a program that simplifies an input stream before sending it to a parser program generated by the **yacc** command.

The function **yylex** analyzes the input stream using a program structure called a *finite state machine*. This structure allows the program to exist in only one state (or condition) at a time. There is a finite number of states allowed. The rules in *file* determine how the program moves from one state to another.

If you do not specify a *file*, **lex** reads standard input. It treats multiple files as a single file.

Note: Since **lex** uses fixed names for intermediate and output files, you can have only one **lex**-generated program in a given directory.

Input File Format (*file*)

The input file can contain three sections: definitions, rules, and user subroutines. Each section must be separated from the others by a line containing only the delimiter, `%%`. The format is:

```

definitions
%%
rules
%%
user subroutines

```

The purpose and format of each are described in the following sections.

Definitions

If you want to use variables in your rules, you must define them in this section. The variables make up the left column, and their definitions make up the right column. For example, if you want to define **D** as a numerical digit, you would write;

```
D    [0-9]
```

You can use a defined variable in the rules section by enclosing the variable name in braces (`{D}`).

In the definitions section, you can set table sizes for the resulting finite state machine. The default sizes are large enough for small programs. You may want to set larger sizes for more complex programs.

```

%p n      Number of positions is n (default 2000)
%n n      Number of states is n (default 500)
%t n      Number of parse tree nodes is n (default 1000)
%a n      Number of transitions is n (default 3000)

```

If extended characters appear in regular expression strings, you may need to reset the output array size with the `%o` parameter (possibly to array sizes in the range 10,000 to 20,000). This reset reflects the much larger number of characters relative to the number of ASCII characters.

Rules

Once you have defined your terms, you can write the rules section. It contains strings and expressions to be matched in *file* to **yylex**, and C commands to execute when a match is made. This section is required, and it must be preceded by the delimiter **%%**, whether or not you have a definitions section. The **lex** command does not recognize your rules without this delimiter.

In this section, the left column contains the pattern to be recognized in an input file to **yylex**. The right column contains the C program fragment executed when that pattern is recognized.

Patterns can include extended characters with one exception: these characters cannot appear in range specifications within character class expressions surrounded by square brackets. The columns are separated by a tab. For example, if you want to search files for the keyword **KEY**, you might write:

```
(KEY)
printf("found KEY");
```

Japanese Language Support Information

Patterns can include SJIS characters, with the same exception as previously noted: the SJIS characters cannot appear in range specifications within character class expressions enclosed in square brackets.

If you include this rule in *file*, the lexical analyzer **yylex** matches the pattern **KEY** and runs the **printf** command.

Each pattern can have a corresponding action, that is, a C command to execute when the pattern is matched. Each statement must end with a semicolon. If you use more than one statement in an action, you must enclose all of them in braces. A second delimiter, **%%**, must follow the rules section if you have a user subroutine section.

When **yylex** matches a string in the input stream, it copies the matched file to an external character array, **yytext**, before it executes any commands in the rules section.

You can use the following operators to form patterns that you want to match:

- | | |
|------------|---|
| x | Matches the character written. x matches the literal character x . |
| [] | Matches any one character in the enclosed range [.-.] or the enclosed list [...] .
[a,b,c,x-z] matches a,b,c,x,y , or z . |
| " " | Matches the enclosed character or string even if it is an operator. "\$" prevents lex from interpreting the character \$ as an operator. |

\	Acts the same as " ". \\$ also prevents the shell from interpreting the character \$ as an operator.
*	Matches zero or more occurrences of the character immediately preceding it. x* matches zero or more repeated.
+	Matches one or more occurrences of the character immediately preceding it.
?	Matches either zero or one occurrences of the character immediately preceding it.
^	Matches the character only at the beginning of a line. ^x matches an x at the beginning of a line.
[^]	Matches any character except the one following the ^. [^x] matches any character except x.

Japanese Language Support Information

The character following the ^ *cannot* be a 2-byte character.

.	Matches any character except the new-line character.
\$	Matches the end of a line.
	Matches either of two characters. x y matches either x or y.
/	Matches one character only when followed by a second character. It reads only the first character into yytext . x/y matches x when it is followed by y, and reads x into yytext .
()	Matches the pattern in the parentheses. This is used for grouping. It reads the whole pattern into yytext . A group in parentheses can be used in place of any single character in any other pattern. (xyz123) matches the pattern xyz123 and reads the whole string into yytext .
{ }	Matches the character as you defined it in the definitions section. If you defined D to be numerical digits, {D} matches all numerical digits.
{m,n}	Matches m to n occurrences of the character. x{2,4} matches 2, 3, or 4 occurrences of x.

If a line begins with only a blank, **lex** copies it to the output file, **lex.yy.c**. If the line is in the declarations section of *file*, **lex** copies it to the declarations section of **lex.yy.c**. If the line is in the rules section, **lex** copies it to the program code section of **lex.yy.c**.

User Subroutines

The **lex** library has three subroutines defined as macros, which you can use in the rules.

input() Reads a character from **yyin**.
unput() Replaces a character after it has been read.
output() Writes an output character to **yyout**.

You can override these three macros by writing your own code for these routines in the user subroutines section. But if you write your own, you must undefine these macros in the definition section as follows:

```
%{  
#undef input  
#undef unput  
#undef output  
}%
```

There is no **main()** in **lex.yy.c** because the **lex** library contains the **main()** that calls **yylex**. Therefore, if you do not include **main()** in the user subroutines section, when you compile **lex.yy.c**, you must enter **cc -ll lex.yy.c**, where **ll** will call the **lex** library.

External names generated by **lex** all begin with the preface **yy**, as in **yyin**, **yyout**, **yylex**, and **yytext**.

Flags

- n Suppresses the statistics summary. When you set your own table sizes for the finite state machine (see page 563), the **lex** automatically produces this summary if you do not select this flag.
- t Writes **lex.yy.c** to standard output instead of to a file.
- v Provides a one-line summary of the generated finite-state-machine statistics.

Files

/usr/lib/libl.a Run-time library.

Related Information

The following command: “**yacc**” on page 1237.

The description of **lex** in *AIX Operating System Programming Tools and Interfaces*.

“Overview of International Character Support” in *Managing the AIX Operating System*.

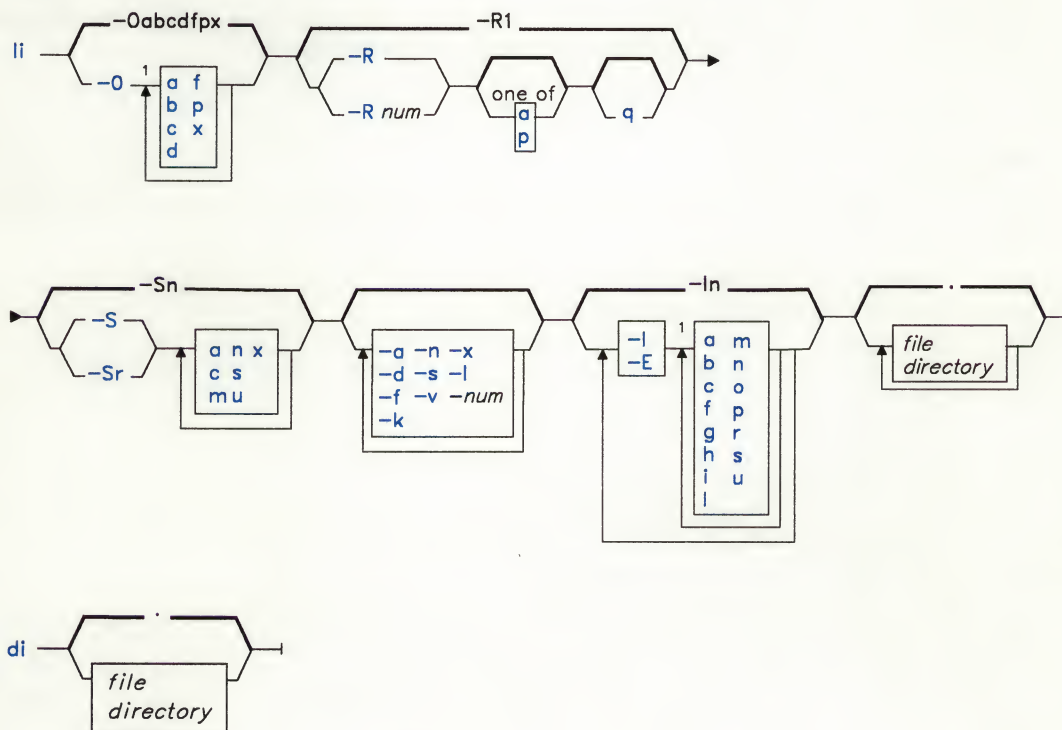
The discussion of Japanese Language Support in *Japanese Language Support User's Guide*.

li

Purpose

Lists the contents of a directory.

Syntax



OL805372

OL805346

OL805308

¹ Do not put a blank between these items.

Description

The **li** command lists the contents of each named *directory* or archive *file* on standard output. For each nonarchive *file* named, **li** displays the file name and any information requested. If you do not specify a *file* or *directory*, **li** lists the current directory.

By default, **li** sorts the output alphabetically and lists it in multiple columns. The collating sequence is determined by the **NLCTAB** environment variable (see “**ctab**” on page 257). It displays control characters in file names in expanded form (for example, `^D`, `\177`). When you specify more than one file or directory, **li** sorts them appropriately, but files appear before directories and their contents. When the date and time appear, the **NLLDATE** and **NLTIME** environment variables control their format. The **NLSMONTH** environment variable controls the short names for months.

The **di** command is equivalent to `li -la lmops`.

Permissions Field

The permissions field displayed with the **-lp** flag contains 11 characters. The first character is:

- d** The entry is a directory.
- b** The entry is a block-type special file.
- c** The entry is a character-type special file.
- l** The entry is a symbolic link.
- p** The entry is a pipe (FIFO).
- The entry is an ordinary file.
- D** The entry is a remote directory.
- F** The entry is a remote ordinary file.
- B** The entry is a remote block special file.
- C** The entry is a remote character special file.
- L** The entry is a remote symbolic link.
- P** The entry is a remote first-in first-out (FIFO) special file.

The next nine characters are interpreted as three sets of 3 bits each. The first set refers to owner permissions, the next to permissions for others in the same group, and the last to all others. Each of the three characters within each set indicate, respectively, permission to read, write, or execute the file. For a directory, execute permission is interpreted as permission to search the directory for a specified file. These permissions are indicated as follows:

- r** If the file is readable.
- w** If the file is writable.
- x** If the file is executable.
- If the corresponding permission is not granted.

The group-execute permission is given as **s** if the file has set-group-ID mode. The user-execute permission character is given as **s** if the file has set-user-ID mode. (For a discussion of these modes, see “**chmod**” on page 160.)

The last character of the field is normally blank, but is displayed as **t** if the 1000 bit of the mode is on. (See the **chmod** system call in *AIX Operating System Technical Reference* for the current meaning of this mode.)

Note: Some combinations of flags do not work well together. For example, **li -vRa** looks unusual, and **li -RSx** and **li -Sx *** are both nearly unintelligible if there are subdirectories contained in the current directory, due to confusion about what level is being listed.

Flags

Flags are grouped into five classes, four of which are always introduced by an uppercase letter: fields (**I** or **E**), restrictions (**O**), recursion (**R**), sort orders (**S**), and miscellaneous. The following flags modify the action of **li**:

-I [hiplogcsmaunrfb]

-E [hiplogcsmaunrfb]

Requests the inclusion (**-I**) or exclusion (**-E**) of certain fields. These fields are selected by the flags in the subset **hiplogcsmaunrfb**. **-I** includes and **-E** excludes the selected fields in the order in which they appear in the argument list. For example, **-l -Ep** excludes the protections field, while **-Ep -l** includes it, since **-l** (the equivalent of **-Icglmop**) follows **-Ep**.

The only field included by default is the name (**n**) field. If you include any other fields, **li** lists the output in single-column rather than multiple-column format. **li** lists the following fields in the following order:

h	Headers
i	I-number
p	Protections
l	Link count
o	Local owner (name or UID)
g	Local group (name or GID)
c	Character count
s	Size in blocks
m	Modified time
a	Accessed time
u	Updated (i-node modified) time
n	Name
r	Node where the entry resides
f	Raw UID of the entry's owner
b	Raw GID of the entry's group.

If the file is a special file, the size (**s**) field contains the major-and minor-device numbers. If you select the **c** (character count) or **s** (size in blocks) flags, **li** writes a total number of blocks for each directory and a grand total when appropriate.

For remote files and directories, the local owner and local group are obtained by using inverse IDs. If there is no inverse ID or if **li** cannot determine the inverse ID, a - (minus sign) displays in the corresponding field. If possible, remote nodes are identified with nicknames. Otherwise, they are identified by their NID displayed in hexadecimal. (See "Distributed Services Concepts" in *Managing the AIX Operating System*.)

For local files and directories that do not have a nickname defined for the local node ID, the node ID field displays as a - (minus sign), and the raw UID (GID) field contains the local owner UID (group GID).

-L Follows a symbolic link and reports on the file at the end of the link.

-O [abcdfp_x]

Requests that the listing be restricted to files of certain types. These types are selected from the subset **abcdfp_x**. The possible types are:

- a** Archives
- b** Block devices
- c** Character devices
- d** Directories
- f** Files (normal, not special)
- p** Pipes (FIFOs)
- x** Executable files (any file with execute permission)

-R[num]apq

Lists recursively to *num* levels deep. The default depth is infinite. This normally displays a single column, with a two-column indentation for each level of the directory structure. When **li** reaches a directory with no subdirectories, it lists the contents of that directory in multiple-column form. Specifying either **-Ra** or **-Rp** suppresses the indentation and multiple-column display. These flags display either the full (**-Ra**) or relative (**-Rp**) path names of each file found. The **-Rq** flag also lists the contents of archive files. When using the **-Rq** flag to list the contents of remote archive files, the user and group fields display as a - (minus sign) unless the **-k** flag is specified. With the **-k** flag, the user and group fields for archive entries display as raw as found in the archive. (See the archive file format in *AIX Operating System Technical Reference*.)

-S [acmnr_sux]

Describes the order in which the listing is to be displayed. The default order is by name (**n**). The **-S_x** flag specifies no sorting. Choosing a flag from the subset **acmnr_sux** selects which field the listing will be sorted by:

- a** Accessed time, latest first
- c** Character count, largest first

- m** Modified time, latest first
- n** Name
- s** Size (same as character count)
- u** Updated time, latest first

If you include the **r** flag with any of these, **li** reverses the order of the sort.

The miscellaneous flags are:

- a** Lists all entries, including those beginning with . (dot).
- d** Lists only the name, not the contents, of directories.
- f** Forces **li** to interpret each *file* as a directory and to list the name found in each slot. All flags requiring information not found in directory entries are turned off and the **-a** flag is turned on. Names are listed in the order that they appear in the directory.
- k** Provides a listing that is equivalent to **li -lbcfmpr**. That is, it lists the permission code, node ID, remote UID, remote GID, time of last modification, character count, and file name for remote entries.
- l** Uses a listing that is equivalent to **li -l cglmop** (the long form listing). That is, it lists the permission code, link count, owner, group, character count, time of last modification, time of last access, and name of each file.

Note: A symbolically linked file is followed by an **→** and then the contents of the symbolic link.
- n** Inhibits the interpretation of control characters in file names. This flag is useful for generating lists of file names for program input or for editing into per-file commands.
- s** Provides a listing similar to that of the **-v** flag, except that the distinguishing marks for file types do not affect sorting (a sortable verbose list). Subdirectories appear in the listing as *name/*, files with execute permission as *name**, special files as *name?*, and symbolic links as *name@*.
- v** Lists files in a way that visually differentiates file types (a verbose visual listing). With this flag, **li** lists subdirectories as [*name*], files with execute permission as <*name*>, special files as **name**, and symbolic links as *@name@*. This differentiation occurs before the **-S** sort. Thus, different types of files are sorted into different parts of the listing.
- x** Displays every available field except headers (an extended form listing). This is equivalent to specifying **li -labcfglimopr su**.
- num** Lists with a maximum of *num* columns. If *num* is unreasonable, **li** picks its own *num*. This flag can be used as in **li -l 1** to make shell files or **li -l 09** to force **li** to display its output in multiple columns. A number appearing in any flag argument is assumed to be the number of columns unless it follows the **-R** flag.

Examples

1. To list the files in the current directory in alphabetical order:

```
li
```

2. To list all files in the current directory, including those with names beginning with a . (dot):

```
li -a
```

3. To display detailed information:

```
li -l chap1 .profile
```

This displays a long listing with detailed information about `chap1` and `.profile`. It lists all the information that you probably need to see. However, `li` can supply even more information with the `-x` flag.

4. To display detailed information about a directory:

```
li -d -l . manual manual/chap1
```

This displays a long listing for the directories `.` and `manual`, and for the file `manual/chap1`. `-d` flag, this would list the files in `.` and `manual` instead of the detailed information about the directories themselves.

5. To list the files in order of modification time:

```
li -Sm -l
```

This displays a long listing of the files that were modified most recently, followed by the older files.

6. To include extra information in the listing:

```
li -lchil
```

In addition to the file name, this lists the character count (`-lc`), i-number (`-li`), and link count (`-ll`) for each file in the current directory. The `-lh` tells `li` to write a heading at the top of each column of information.

7. To list the contents of each directory in a tree:

```
li -R manual
```

This lists the names in each subdirectory of the tree that starts with `manual`.

Files

/etc/passwd	Contains user names for <code>li -Io</code> .
/etc/group	Contains group names for <code>li -Ig</code> .

Related Information

The following commands: “**ctab**” on page 257 and “**ls**” on page 595.

The **chmod** system call and the **environment** miscellaneous facility in *AIX Operating System Technical Reference*.

“Overview of International Character Support” in *Managing the AIX Operating System*.

“Distributed Services Concepts” in *Managing the AIX Operating System*.

line

line

Purpose

Reads one line from the standard input.

Syntax

`line` —

OL805142

Description

The **line** command copies one line from standard input and writes it to standard output. It returns an exit value of 1 on an end-of-file and always writes at least a new-line character. Use this command within a shell command file to read from your work station.

Example

To read a line from the keyboard and append it to a file:

```
echo 'Enter comments for the log:'  
echo ': \c'  
line >>log
```

This shell procedure displays the message:

```
Enter comments for the log:
```

then reads a line of text from the work station keyboard and adds it to the end of log. The `echo ': \c'` command displays a colon prompt. See “**echo**” on page 369 for information about the `\c` escape sequence.

Related Information

The following command: “**sh**” on page 913.

The **read** system call in *AIX Operating System Technical Reference*.

link, unlink

Purpose

Performs a link or unlink system call.

Syntax

`link` — *file1* — *file2* —

OL805143

`unlink` — *file* —

OL805227

Description

The **link** and **unlink** commands perform the corresponding system calls of the same name on the specified file, abandoning all error checking. These commands can be run only by a user operating with superuser authority (see “su” on page 1026). You should be familiar with the **link** and **unlink** system calls described in *AIX Operating System Technical Reference*.

The **link** and **unlink** commands do not issue error messages when the associated system call fails; you must check the exit value to determine if the command completed normally. Each returns a 0 if it succeeds, a 1 if you specify too few or too many parameters, and a 2 if its system call fails.

Warning: The **link** and **unlink** commands allow the superuser to deal with unusual problems, such as moving an entire directory to a different part of the directory tree. They also permit you to create directories that cannot be reached or escaped from. Be careful to preserve directory structure.

link

To preserve directory structure observe the following rules:

- Be certain every directory has a . (dot) link to itself.
- Be certain every directory has a .. (dot dot) link to its parent directory.
- Be certain every directory has no more than one link to it.
- Be certain every directory is accessible from the root of its file system.

Note: If the . (dot) entry has been destroyed and **fsck** is unable to repair it (a rare occurrence), you can use the **link** command to restore the . (dot) entry of the damaged directory with the command: **link dir dir/**. where *dir* is the name of the damaged directory. However, use this only as a last resort when the directory is destroyed and **fsck** is unable to fix it.

Related Information

The following commands: “**ln**” on page 581 and “**fsck**, **dfsc**” on page 445.

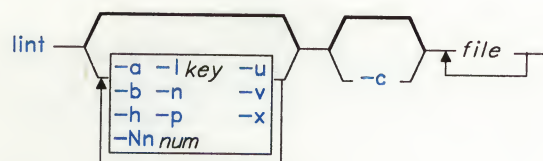
The **link** and **unlink** system calls in *AIX Operating System Technical Reference*.

lint

Purpose

Checks C programs for potential problems.

Syntax



OL805433

Description

The **lint** program checks C language source code for coding and syntax errors and for inefficient or nonportable code. You can use this program to:

- Identify source code and library incompatibility
- Enforce type checking rules more strictly than does the compiler
- Identify potential problems with variables
- Identify potential problems with functions
- Identify problems with flow control
- Identify legal constructions that may produce errors or be inefficient
- Identify possibly nonportable code.

The **lint** command assumes that *file* names ending in **.c** are C Language source files. It assumes that those ending in **.ln** are the result of an earlier running of **lint** with either the **-c** or the **-o** flag used. These **.ln** files are analogous to the **.o** (object) files produced by the **cc** command when given a **.c** file as input. **lint** warns you about files with other suffixes and ignores them.

The **lint** command takes all the **.c** and **.ln** files and the libraries specified by **-l** flags and processes them in the order that they appear on the command line. By default, it adds the standard **lint** library (**llib-lc.ln**) to the end of the list of files. However, when you select the **-p** flag, **lint** uses the portable library **llib-port.ln**. By default, the second pass of **lint** checks this list of files for mutual compatibility; however, if you specify the **-c** flag, **lint** ignores the **.ln** and **lib-lx** files.

lint

The **-c** and **-o** flags allow for incremental use of **lint** on a set of C Language source files. Generally, you use **lint** once for each source file with the **-c** flag. Each of these runs produces a **.ln** file that corresponds to the **.c** file and writes all messages that are about just that source file. After you have run all source files separately through **lint**, you run it once more, without the **-c** flag, listing all the **.ln** files with the needed **-l** arguments. This writes all interfile inconsistencies. This procedure works well with the **make** command, allowing it to run **lint** on only those source files that have been modified since the last time that set of source files was checked.

The following comments in a C source program change the way that **lint** operates when checking the source program:

- /*NOTREACHED*/** Suppresses comments about unreachable code.
- /*VARARGS*n**/** Suppresses checking the following function declaration for varying numbers of arguments but does check the data type of the first *n* arguments. If you do not include a value for *n*, **lint** checks no arguments (*n*=0).
- /*ARGSUSED*/** Turns on the **-v** flag for the next function.
- /*LINTLIBRARY*/** If you place this comment at the beginning of a file, **lint** does not identify unused functions in the file.

The **lint** command first writes messages about each source file as it processes the file. It collects messages about included files and writes those after it has gone through all the source files. Finally, if you have not specified the **-c** flag, it collects information gathered from all input files and checks it for consistency. At this point, if it is not clear whether a message stems from a given source file or from one of its included files, **lint** displays the source file name followed by a question mark.

Flags

- a** Suppresses messages about assignments of long values to variables that are not long.
- b** Suppresses messages about unreachable break statements.
- h** Does not try to detect bugs, improve style, or reduce waste.
- c** Causes **lint** to produce a **.ln** file for every **.c** file on the command line. These **.ln** files are the product of the first pass of **lint** only and are not checked for interfunction compatibility.
- l*key*** Includes the additional **lint** library **llib-l*key*.ln**. You can include a **lint** version of the math library **llib-lm.ln** by specifying **-lm** on the command line or **llib-ldos.ln** by specifying **-ldos** on the command line. Use this flag to include local **lint** libraries when checking files that are part of a project having a large number of files. This flag does not prevent **lint** from using the **llib-lc.ln** library.

-n	Does not check for compatibility with either the standard or the portable lint libraries.
-Nnnum	Increases the size of the symbol table. The default size is 1500.
-o lib	Causes lint to create a lint library with the name llib-llib.ln . The -c flag nullifies any use of the -o flag. The lint library produced is the input that is given to the second pass of lint . The -o flag simply causes this file to be saved in the named lint library. To produce a llib-llib.ln without extraneous messages, use the -x flag. The -v flag is useful if the source files for the lint library are just external interfaces (for example, the way the file llib-lc is written). These flag settings are also available through the use of lint comment lines.
-p	Checks for portability to other C dialects.
-u	Suppresses messages about functions and external variables that are either used and not defined or defined and not used. Use this flag to run lint on a subset of files of a larger program.
-v	Suppresses messages about function parameters that are not used.
-x	Suppresses messages about variables that have external declarations but are never used.

In addition, **lint** recognizes the following flags of the **cpp** command (macro preprocessor):

-Dname[=def]	Defines the name , as if by #define . The default <i>def</i> is 1.
-Idir	Adds <i>dir</i> to the list of directories in which lint searches for #include files.
-Uname	Removes any initial definition of name , where name is a reserved symbol that is predefined by the particular preprocessor.

Examples

1. To check a C program for errors:
`lint program.c`
2. To suppress some of the messages:
`lint -v -x program.c`

This checks `program.c`, but does not display error messages about unused function parameters (**-v**) or unused externals (**-x**).

lint

3. To check the program against an additional lint library:

```
lint -lsubs program.c
```

This checks program.c against both the standard lint library (/usr/lib/lldb-lc.ln) and /usr/lib/lldb-lsubs.ln.

4. To check against the portable library and an additional library:

```
lint -lsubs -p program.c
```

This checks program.c against both the portable lint library (/usr/lib/lldb-port.ln) and /usr/lib/lldb-lsubs.ln.

5. To check against a nonstandard library only:

```
lint -lsubs -n program.c
```

This checks program.c against only /usr/lib/lldb-lsubs.ln.

Files

/usr/lib/lint[12]	Programs.
/usr/lib/lldb-lc.ln	Declarations for standard functions (binary format).
/usr/lib/lldb-lc	Declarations for standard functions (source).
/usr/lib/lldb-port.ln	Declarations for portable functions (binary format).
/usr/lib/lldb-port	Declarations for portable functions (source).
/usr/lib/lldb-lm.ln	Declarations for standard math functions (binary format)
/usr/lib/lldb-lm	Declarations for standard math functions (source)
/usr/lib/lldb-ldos.ln	Declarations for shell functions (binary format).
/usr/lib/lldb-ldos	Declarations for shell functions (source).
/usr/tmp/*lint*	Temporary files.

Related Information

The following command: “cc” on page 140.

The topic “Checking C Programs” in *AIX Operating System Programming Tools and Interfaces*.

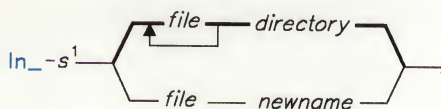
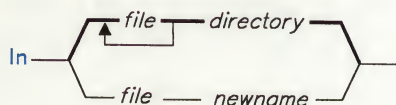
“Overview of International Character Support” in *Managing the AIX Operating System*.

ln

Purpose

Links files.

Syntax



¹This flag is not supported by Japanese Language Support

OL805028

Description

The **ln** command links *file* to *newname* (in the current directory), or to the same name (*file*) in another existing *directory*. You can link directories, provided the two directories have the same parent.

If you are linking a file to a new name, you can list only one *file*. If you are linking to a *directory*, you can list more than one *file*.

Note: You can only link files across file systems using symbolic links.

Flags

- s Creates a symbolic link to a file or directory. Specifying the complete path name of the file or directory is recommended.

Examples

1. To create another name (also called an alias) for a file:

```
ln chap1 intro
```

This links chap1 to the new name intro. If intro does not already exist, the file name is created. If intro does exist, the file is replaced by a link to chap1. Now chap1 and intro are two file names that refer to the same file. Any changes made to one also appear in the other. If one name is deleted with **del** or **rm**, the file is not actually deleted, but remains under the other name.

2. To link a file to the same name in another directory:

```
ln index manual
```

This links index to the new name manual/index.

Note the difference: intro in Example 1 is the name of a file; manual in Example 2 is a directory that already exists.

3. To link several files to names in another directory:

```
ln chap2 jim/chap3 /u/manual
```

This links chap2 to the new name /u/manual/chap2 and jim/chap3 to /u/manual/chap3.

4. To use **ln** with pattern-matching characters:

```
ln manual/* .
```

This links all files in the directory manual into the current directory (.), giving them the same names they have in manual. Note that you must type a space between the asterisk and the period.

Related Information

The following commands: “**rm**” on page 833, “**mv**” on page 679, and “**cp**” on page 202.

The **chmod** and **link** system calls in *AIX Operating System Technical Reference*.

The symbolic link section in *Using the AIX Operating System*.

locator

Purpose

Controls the sample rate of the locator.

Syntax

`locator — -rate —`

OL805444

Description

The **locator** command sets the *rate* at which the system checks, per second, the cursor position controlled by the mouse. You can specify any of the following *rates*: **10**, **20**, **40**, **60**, **80**, or **100**. Initially, at system startup, this rate is set at **60**.

Note: You can run the **locator** command only from the system console.

Flag

-rrate Sets the sampling *rate* to the specified value.

Example

To set the locator rate to **40**:

```
locator -r40
```


login

login

Purpose

Allows you to sign on to the system and performs user identification and authentication.

Syntax

`login -r node 1`

¹ This command is not normally entered on the command line

OL805005

Description

The **login** program logs you in to the system and performs user authentication. Its primary functions are the following:

- Identify the user and validates the user's password
- Make the required audit, accounting, and log entries
- Execute **loginx** or **passwd**.

A **logger process**, initially running the **getty** program, is started for each enabled port. The **getty** command reads a login name and sets work station modes (see "**getty**" on page 490). Then it runs **login**, which may ask for a password. If you do not have a password, press the **Enter** key.

Your login attempt might fail for the following reasons:

- Your login name/password pair does not match an entry in the password file.
- Your password has expired. This can happen if your system requires that you change your password after a set number of days. In this case, **login** runs the **passwd** command instead of letting you log in. (For more information, see "**passwd**" on page 735.) After you change your password, you can attempt to log in again.
- The system has reached the limit of simultaneously logged-in users. Each AIX kernel sets a limit on the number of concurrent logins by nonprivileged users; this limit may be one. A **privileged** user is one that has a user ID from 0 to 20. A privileged user can log in at any time.

- Your account is no longer active. The administrator of the system has invalidated your account by specifying the **nouse** value for the **restrictions** attribute in the **/etc/security/passwd** file. This failure results in the message: **You cannot login with this account..**
- You are not allowed to login with this account. The administrator of the system has prevented logins to your account by specifying the **nologin** value for the **restrictions** attribute in the **/etc/security/passwd** file. This failure results in the message: **You cannot login with this account..**

In one special case, **login** does not ask for a user name and password pair. When the login port is the console and the file **/etc/autolog** contains a valid user name, **login** creates a login session for that user automatically. Other processing by **login** proceeds normally.

When a user logs in successfully, the **login** program makes entries in **/etc/utmp**, the record of users logged in to the system, and in **/usr/adm/wtmp** (if it exists), for use in accounting. On invalid login attempts (due to incorrect login names or passwords), **login** makes entries in the **/etc/.ilog** file.

When you log in as user **root** or **su** and the **/etc/.ilog** file is not empty, you see a message advising you to check the **/etc/.ilog** file for a record of unsuccessful login attempts.

Environment variables inherited from **getty** and **init** (such as those specified in **/etc/environment**) are kept. You may expand or modify the environment by supplying additional parameters to **login** when it requests your login name. These may take the form **xxx** or **xxx=yyy**. Parameters without an equal sign are placed in the environment as **Lnum=xxx**, where **num** is a number starting at 0 and incremented each time a new variable name is required. Parameters containing an equal sign are placed into the environment without modification. If they already exist, the new assignment replaces the older value. However, you cannot change the shell variables **PATH** and **SHELL**. (This restriction prevents people who log in to restricted environments from creating unrestricted secondary shells.)

Flags

-rnode Identifies the login as a remote login and specifies the node requesting the login.

Files

/etc/utmp	Accounting file.
/usr/adm/wtmp	Accounting file.
/etc/.ilog	Accounting file.
/etc/autolog	Login ID for automatic login.
/etc/passwd	Password file.
/etc/security/config	File containing security-relevant information.
/etc/security/passwd	File containing password information.

Related Information

The following commands: “**users**, **adduser**” on page 1129, “**cs****h**” on page 225, “**getty**” on page 490, “**init**” on page 521, “**passwd**” on page 735, and “**pstart**, **penable**, **pshare**, **pdelay**” on page 791.

Note: The **cs****h** command contains a built-in subcommand named **login**. The command and subcommand do not necessarily work the same way. For information on the subcommand, see the **cs****h** command.

The **passwd** and **utmp** files in *AIX Operating System Technical Reference*.

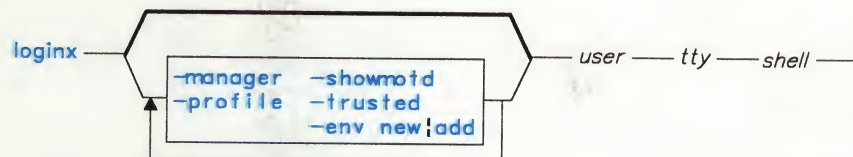
The discussion of login sessions in *Managing the AIX Operating System*.

loginx

Purpose

Sets up a user's execution environment.

Syntax



A5ACG022

Description

This command is called by **login** and **su**. The **loginx** command sets up your execution environment and enhances the environment according to the flags you select. It then runs the specified shell on the specified terminal.

This program always sets the uid and gid to that of the specified user. It sets the user's audit classes as defined in **/etc/security/passwd** as well as the file size limit for this user. In addition, it sets the HOME, LOGNAME, PATH, MAIL, and SHELL environment variables.

Japanese Language Support Information

If Japanese Language Support is installed on your system, this command is not available.

Flags

- manager** The **-manager** flag causes this command to read the **/etc/ports** file for the **shell** attribute and executes that value. The **shell** given on the command line becomes the first parameter to the attribute value.
- profile** The **-profile** flag causes **loginx** to change the current directory to the user's home directory as specified in **/etc/passwd**. When the user's shell is run, its name is preceded by a - (minus) sign.
- showmotd** The **-showmotd** flag causes **/etc/motd** to be displayed on the screen.

loginx

- trusted** The **-trusted** flag causes the terminal to become untrusted and the terminal mode and ownership to change. The mode changes to 0600, and the ownership changes to that of the specified user.
- env new!add** The **-env** flag must be followed by either **new** or **add**. If **new** is specified, the current shell environment variables are cleared. If **add** is specified, the current shell environment variables are preserved.

Files

- /etc/ports** Contains the names and characteristics of the system terminal ports.
- /etc/security/passwd** Contains password information necessary for security.

Related Information

The following commands: “**login**” on page 584, “**su**” on page 1026, “**shell**” on page 938, “**tty**” on page 1105, and “**users, adduser**” on page 1129

The **/etc/passwd** and **/etc/ports** files in *AIX Operating System Technical Reference*.

The discussion of login sessions in *Managing the AIX Operating System*.

logname

Purpose

Displays your login name.

Syntax

logname —

OL805145

Description

The **logname** command writes to standard output the name you used to log in to the system. It is the contents of the environment variable **\$LOGNAME**, which is set when you log in to the system.

Files

/etc/profile System profile.

Related Information

The following commands: “**env**” on page 393 and “**login**” on page 584.

The **logname** subroutine *AIX Operating System Technical Reference*.

The **environ** special facility in *AIX Operating System Technical Reference*.

logout

logout

Purpose

Stops all processes on a port, returning it to a dead state.

Syntax

`logout` —

OL805485

Description

The **logout** command provides a thorough method of logging off your system. The command verifies that the user invoking **logout** logged in at the same port. If the **login** user and the **logout** user do not match, **logout** permission is denied, and the command stops.

Note: Only the superuser can redirect standard input for this command. The **logout** command is a setuid-root program. It sends a **SIGKILL** to the process group leader running on the port from which the **logout** command is entered. This command also issues the **revoke()** system call for the **logout** port and any synonym of that port.

Japanese Language Support Information

If Japanese Language Support is installed on your system, this command is not available.

Files

<code>/etc/utmp</code>	Contains a record of logged-in users.
<code>/etc/ports</code>	Specifies the port synonym.

Related Information

The following commands: “**login**” on page 584 and “**tsh**” on page 1100.

The `/etc/ports` and the `/etc/utmp` file formats in *AIX Operating System Technical Reference*.

lorder

Purpose

Finds the best order for member files in an object library.

Syntax

`lorder` 

OL805029

Description

The **lorder** command reads one or more object or library archive *files*, looking for external references and writing a list of paired file names to standard output. The first of each paired files contains references to identifiers that are defined in the second file. You can send this list to the **tsort** command to find an ordering of a library member file suitable for one-pass access by **ld**.

If object files do not end with **.o**, **lorder** overlooks them and attributes their global symbols and references to some other file.

Example

To create a subroutine library:

```
lorder charin.o scanfld.o scan.o scanln.o | tsort | xargs ar qv libsubs.a
```

This creates a subroutine library named `libsubs.a` that contains `charin.o`, `scanfld.o`, `scan.o`, and `scanln.o`. The ordering of the object modules in the library is important. The **ld** command requires each module to precede all the other modules that it calls or references. The **lorder** and **tsort** commands together add the subroutines to the library in the proper order.

lorder

Suppose that `scan.o` calls `scanfld.o` and `scanln.o`. `scanfld.o` also calls `charin.o`. First, the **lorder** command creates a list of pairs that shows these dependencies:

```
charin.o charin.o
scanfld.o scanfld.o
scan.o scan.o
scanln.o scanln.o
scanfld.o charin.o
scanln.o charin.o
scan.o scanfld.o
```

Next, the `l` (vertical bar) sends this list to the **tsort** command, which converts it into the ordering we need:

```
scan.o
scanfld.o
scanln.o
charin.o
```

Note that each module precedes the module it calls. `charin.o`, which does not call another module, is last.

The second `l` then sends this list to **xargs**, which constructs and runs the following **ar** command:

```
ar qv libsubs.a scan.o scanfld.o scanln.o charin.o
```

This **ar** command creates the properly ordered library.

Files

`/tmp/sym*` Temporary files.

Related Information

The following commands: “**ar**” on page 55, “**ld**” on page 557, “**nm**” on page 705, “**tsort**” on page 1102, and “**xargs**” on page 1232.

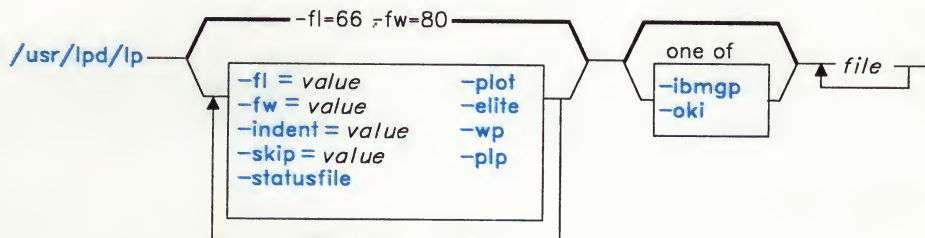
The **ar** file in *AIX Operating System Technical Reference*.

lp

Purpose

Prints a file in a format suitable for sending to a line printer.

Syntax



OL805396

Description

The **lp** command prints *file* on its standard output in a form that is suitable for a line printer. The **lp** command is normally invoked by the **qdaemon** command. **qdaemon** directs the output from **lp** to the appropriate device.

Flags are passed to **lp** in the following ways:

- Flags specified in the **qconfig** structure are passed each time that **lp** is invoked. The **-plp**, **-ibmgrp**, **-oki**, and **-statusfile** flags most likely appear in **qconfig**.
- Flags that are not recognized by the **print** command are assumed to be for **lp** and are passed to **lp** with the requested job.

Flags

-elite	Prints the text at 12 characters per inch instead of 10 characters per inch. This flag changes the default forms width to 96 characters.
-fl = value	Sets the forms length equal to <i>value</i> . The default length is 66 lines.
-fw = value	Sets the forms width equal to <i>value</i> . The default width is 80 columns. Lines that are wider than <i>value</i> are truncated. If you set <i>value</i> to 0, no truncation is performed.
-ibmgrp	Specifies an IBM Graphic Printer.

- indent = *value*** Indents the printed output the number of spaces specified with *value*.
- oki** Specifies an Okidata Model 92 or 93.
- plot** Passes text directly to the printer without processing. This is useful when using the printer as a plotter. Normally, lines that contain backspaces and carriage-return characters are processed so that they print with minimum print head motion. The sequence **ESC-9** maps to half-line feeds.
- plp** Sets and resets printer port parameters, if the printer is attached with a parallel interface.
- skip = *value*** Does not print the first *value* blank lines in the file.
- statusfile** Updates the status information in the status file that is open on file descriptor 3. The status information is passed from **qdaemon**.
- wp** Sets the printer, if possible, to the Word Processing mode.

Related Information

The following commands: “**print**” on page 767 and “**qdaemon**” on page 802.

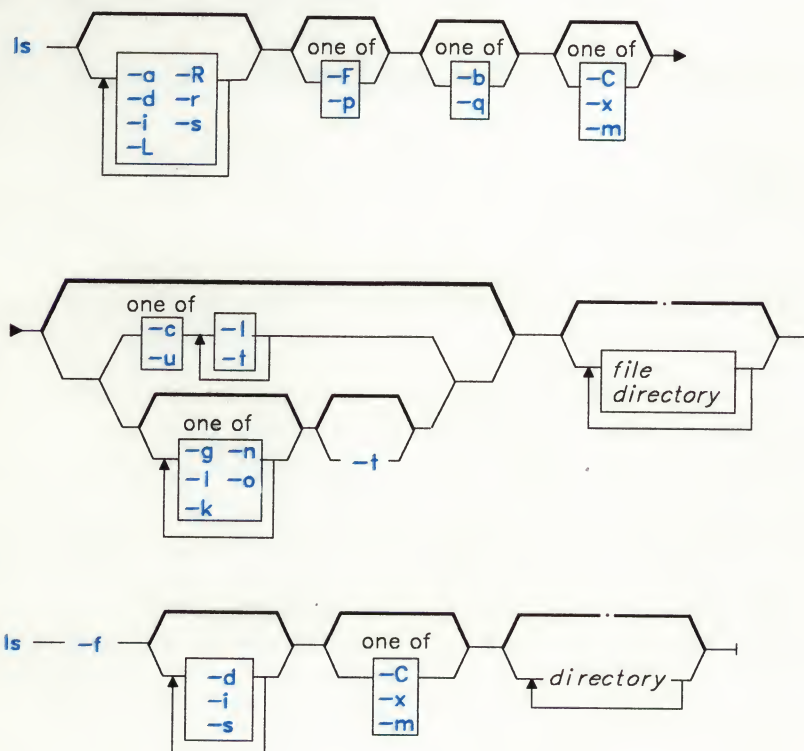
The **qconfig** file in *AIX Operating System Technical Reference*.

ls

Purpose

Displays the contents of a directory.

Syntax



OL805030

OL805243

Description

The `ls` command writes to standard output the contents of each specified *directory* or the name of each specified *file*, along with any other information you ask for with the flags. If you do not specify a *file* or *directory*, `ls` displays the contents of the current directory.

By default, **ls** displays all information in alphabetic order by file name. The collating sequence is determined by the **NLCTAB** environment variable (see “**ctab**” on page 257). Individual file names are listed before directory names.

There are three main ways to format the output:

- List one entry per line. This is the default format.
- List entries in multiple columns by specifying either the **-C** or **-x** flags.
- List entries in a comma-separated series by specifying the **-m** flag.

To determine the number of character positions in the output line, **ls** uses the environment variable **COLUMNS**. If this variable is not set, it reads the **terminfo** file. If **ls** cannot determine the number of character positions by either of these methods, it uses a default value of 80.

The mode displayed with the **-l** flag is interpreted as follows:

If the first character is:

- d** The entry is a directory.
- b** The entry is a block special file.
- c** The entry is a character special file.
- l** The entry is a symbolic link.
- p** The entry is a first-in first-out (FIFO) special file.
- The entry is an ordinary file.
- D** The entry is a remote directory.
- F** The entry is a remote ordinary file.
- B** The entry is a remote block special file.
- C** The entry is a remote character special file.
- L** The entry is a remote symbolic link.
- P** The entry is a remote first-in first-out (FIFO) special file.

The next nine characters are divided into three sets of three characters each. The first three characters show the owner's permission. The next set of three characters show the permission of the other users in the group. The last set of three characters show the permission of any one else with access to the file. The three characters in each set show read, write, and execute permission of the file. Execute permission of a directory lets you search a directory for a specified file.

Permissions are indicated as follows:

- r** You can read the file.
- w** You can edit (write) the file.
- x** You can search the file.
- You do not have permission to access the file.

The group-execute permission character is **s** if the file has set-group ID mode. The user-execute permission character is **S** if the file has set-user-ID mode. The last character of the mode (normally **x** or **-**) is **t** if the 1000 (octal) bit of the mode is set; see “**chmod**” on

page 160 for the meaning of this mode. The indications of set-ID and 1000 bit of the mode are capitalized (*S* and *T* respectively) if the corresponding execute permission is not set.

When the size of the files in a directory are listed, the **ls** command displays a total count of blocks, including indirect blocks.

The environment variables **NLLDATE** and **NLTIME** control the format of the date and time. The environment variable **NLSMONTH** controls the short names of months.

Flags

- a Lists all entries in the directory including the entries that begin with a . (dot).
- b Displays nonprintable characters in an octal \nnn notation.
- c Uses the time of last modification of the i-node (file created, mode changed, and so on) for sorting (when used with -t) or for displaying (when used with -l). This flag has no effect when not used with either -t or -l or both.
- C Sorts output vertically in a multicolumn format.
- d Displays only the information for the directory named. This is useful with the -l flag to get the status of a directory.
- f Lists the name in each slot for each named *directory*. This flag turns off -l, -t, -s, and -r, and turns on -a; the order is the order in which entries appear in the directory.
- F Puts a / (slash) after each file name if the file is a directory, an * (asterisk) after each file name if the file can be executed, and an @ sign after each file name if the file is a symbolic link.
- g Displays the same information as with -l, except for the owner.
- i Displays the i-number in the first column of the report for each file.
- k Displays the permission codes, node ID, remote UID, remote GID, time of last modification, size (in bytes), and file name for remote entries.

For remote files and directories, the local owner and local group are obtained by using inverse IDs. If there is no inverse ID or if **ls** cannot determine the inverse ID, a - (minus sign) displays in the corresponding field. If possible, remote nodes are identified with nicknames. Otherwise, they are identified by their NID displayed in hexadecimal. (See "Distributed Services Concepts" in *Managing the AIX Operating System*.)

For local files and directories that do not have a nickname defined for the local node ID, the node ID field displays as a - (minus sign), and the raw UID (GID) field contains the local owner UID (group GID).

- l Displays the mode, number of links, owner, group, size (in bytes), and time of last modification for each file. If the file is a special file, the size field will instead contain the major and minor device numbers.

Note: A symbolically linked file is followed by an \rightarrow and then the contents of the symbolic link.

- L Follows a symbolic link and reports on the file at the end of the link.
- m Uses stream output format (a comma-separated series).
- n Displays the same information as with -l, except that it displays the user and the group IDs instead of the user and group names.
- o Displays the same information as with -l, except for the group.
- p Puts a slash after each file name if that file is a directory. This is useful when you pipe the output of **ls** to the **pr** command as follows:

```
ls -p | pr -5 -t -w80
```
- q Displays nonprintable characters in file names as the character `?`.
- r Reverses the order of the sort, giving reverse alphabetic or the oldest first, as appropriate.
- R Lists all subdirectories recursively.
- s Gives size in blocks (including indirect blocks) for each entry.
- t Sorts by time of last modification (latest first) instead of by name.
- u Uses the time of the last access instead of time of the last modification for sorting (when used with -t) or for displaying (when used with -l). This flag has no effect when not used with either -t or -l or both.
- x Sorts output horizontally in a multicolumn format.

Examples

1. To list all files in the current directory:

```
ls -a
```

This lists all files, including `.` (dot), `..` (dot-dot), and other files with names beginning with a dot.

2. To display detailed information:

```
ls -l chap1 .profile
```

This displays a long listing with detailed information about `chap1` and `.profile`.

3. To display detailed information about a directory:

```
ls -d -l . manual manual/chap1
```

This displays a long listing for the directories `.` and `manual`, and for the file `manual/chap1`. Without the `-d` flag, this would list the files in `.` and `manual` instead of the detailed information about the directories themselves.

4. To list the files in order of modification time:

```
ls -l -t
```

This displays a long listing of the files that were modified most recently, followed by the older files.

Files

<code>/etc/passwd</code>	Contains user IDs.
<code>/etc/group</code>	Contains group IDs.
<code>/usr/lib/terminfo/*</code>	Contains terminal information.

Related Information

The following commands: “**chmod**” on page 160, “**ctab**” on page 257, and “**find**” on page 422.

The **environment** miscellaneous facility in *AIX Operating System Technical Reference*.

“Overview of International Character Support” in *Managing the AIX Operating System*.

“Distributed Services Concepts” in *Managing the AIX Operating System*.

The symbolic link section in *Using the AIX Operating System*.

ls
